UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

IDENTIFICACIÓN AUTOMÁTICA DE MATERIALES USANDO EL SENSOR KINECT

Alejandro López Cifuentes
Tutor: Marcos Escudero Viñolo

-TRABAJO FIN DE GRADO-

Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2015

IDENTIFICACIÓN AUTOMÁTICA DE MATERIALES USANDO EL SENSOR KINECT

Alejandro López Cifuentes

Tutor: Marcos Escudero Viñolo

Video Processing and Understanding Lab
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2015

Resumen

Este proyecto desarrollado tiene como objetivos principales tanto el diseño y captura de una base de datos de imágenes como el diseño de un sistema capaz de entrenar un modelo y clasificar imágenes previamente capturadas. La finalidad de este proceso es conseguir desarrollar un prototipo que realice una aproximación a la identificación automática de materiales mediante el sensor Kinect.

Para lograr estos objetivos se ha realizado un estudio previo de las diferentes versiones actuales de Kinect, de los diferentes métodos de caracterización de materiales como aproximaciones a la función *Bidirectional Reflectance Distribution Function* (BRDF) entre los que destacan las aproximaciones en dos dimensiones como Histogram of Oriented Gradients (HoG), así como los distintos métodos de generación de modelos de conocimiento como las *Support Vector Machines* (SVM). Adicionalmente se han estudiado las bases de datos de materiales actualmente disponibles.

Se ha grabado una base de datos mediante Kinect capturando una selección de materiales desde diferentes ángulos de captura e incidencia de iluminaciones. Para añadir información adicional la base de datos contará ademas de con imágenes en color, con imágenes de infrarrojos y de profundidad. Cada imagen de material lleva un parche asociado cuya finalidad será aislar el material del entorno de grabación para su posterior análisis.

Una vez capturada la base de datos se ha diseñado un sistema que extrae las características de los parches previamente definidos mediante los descriptores GDF-HOG y EigHess-HOG. Estos descriptores son extensiones de HoG invariantes a rotaciones. Mediante el uso de SVM se han generado modelos de conocimiento que nos permiten predecir y clasificar muestras de entrada de evaluación del sistema.

Por último se ha evaluado la bondad del sistema desarrollado obteniendo buenos resultados en situaciones controladas y consiguiendo, en imágenes de situaciones reales, resultados prometedores.

Palabras Clave

Reconocimiento automático de materiales, sensor Kinect, modelos de conocimiento, Máquinas de Vector Soporte, Histograma de Gradientes Orientados.

Abstract

The following project has as main objetives to design and capture an image database and a system capable to train a model and classify a set of previously captured images. The aim of this process is to develop a prototype that performs an aproximation to automatic material recognition through the Kinect sensor.

In order to achive these objetives several stages need to be first fulfilled. A study about recent versions of the Kinect sensor is first required. Then, existing methods for material characterization should be reviewed. These can be understood as versions of Bidirectional Reflectance Distribution Function (BRDF). In particular Histogram of Oriented Gradients (HoG) can be seen as a two-dimensional aproximation to BRDF. The generation of knwoledge models via Support Vector Machines (SVM) is also analysed. Finally datasets currently available for research have been also listed.

A new dataset has been recorded using Microsoft Kinect capturing material selection from diferent capture and ilumination incidence angles. In order to include aditional information to colour images, depth and infrared images would be also included in the dataset. Each material image has an associated patch which aim is to isolate the material from its background.

Once the dataset has been recorded a system using GDF-HOG and EigHess-HOG descriptors in order to extract patch characterization has been design. These descriptors are rotation invariant HoG extensions. Through SVM knowledge models have been generated. These models allow to predict and clasify untrained input instances.

Finally system performance has been evaluated achieving good results in supervised situations and promising results in real scenes.

Keywords

Automatic material recognition, Kinect sensor, knowledge models, Support Vector Machines, Histogram of Oriented Gradients.

Agradecimientos

En primer lugar quiero agradecer todo el trabajo y dedicación a mi tutor, Marcos. Gracias por darme la oportunidad de haber conseguido el trabajo que quería, por todo el apoyo que he recibido durante todo el año y por ayudarme hasta el ultimo día para llegar al final. Ha sido un enorme placer.

Igualmente quiero agradecer a todos los compañeros del VPU el hacer del laboratorio un sitio agradable donde ir a trabajar y donde nunca falta el buen ambiente y las risas, así da gusto.

Gracias a todos los compañeros de estos últimos cuatro años, habéis conseguido que se pueda sobrellevar la rutina del día a día haciendo las clases algo mejores gracias al buen rollo. En especial a ese grupo formado en primero de carrera y del cual puedo decir que después de cuatro años me llevo grandes amigos y que espero que lo sigan siendo de aquí en adelante. Gracias Adri, Dani, Javi Gallego, Javi, Zaabi, Jose, Juan e Iñaki. Sois muy grandes.

A Ana, por estar ahí estos cuatro duros años animándome cada día y apoyándome cada vez que se hacia cuesta arriba la carrera, gracias de verdad.

Por último pero no menos importante, agradecer el apoyo continuo de mi familia, de mis padres y mi hermana, por haber hecho de mi la persona que soy, haberme dado la oportunidad de haber estudiado lo que quería y en definitiva por todo el día a día estos últimos 22 años.

Para todos vosotros

Alejandro López Cifuentes.
Julio 2015.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

Podemos definir el reconocimiento de materiales como la capacidad que poseen naturalmente los seres humanos para poder distinguir de que está hecho un objeto, es decir, distinguir si un elemento esta fabricado de un determinado material u otro.

El proceso de reconocimiento de materiales es importante para los seres humanos ya que nos permite hacer predicciones acerca del comportamiento del mundo real y poder tomar decisiones en consecuencia con ellas. Nos permite saber si vamos a resbalar por un suelo mojado dependiendo del tipo de material que se haya usado en su construcción, saber como debemos y con que fuerza agarrar un objeto cuando lo vayamos a coger en función del elemento que se haya necesitado para su fabricación, así como saber si vamos a romper un objeto o simplemente rallarlo durante un uso dependiendo del material con el que se haya realizado.

El diseño de un sistema artificial automático para el reconocimiento de materiales permitiría en teoría traspasar parte de estos procesos de decisión a un sistema artificial. Pongamos el caso de un brazo robótico, que suponiendo un funcionamiento ideal del sistema, podrá tomar las mismas decisiones en cuanto a la presión y fuerza con la que agarra un determinado objeto que un ser humano. Para poder alcanzar este tipo de funcionalidades idílicas es necesario desarrollar previamente un sistema de reconocimiento de materiales que funcione correctamente con independencia de la forma del objeto que compongan, del tipo de iluminación a la que estén sometidos y del punto de vista desde el que esté siendo observado.

El proceso de reconocimiento automático de materiales es una tarea de amplia utilización potencial. El hecho de poder conocer características determinantes de un material, como por ejemplo cómo reflejan la luz, cuáles son sus patrones de reflexión

o cómo es su superficie, permite caracterizarlo y describirlo de forma idónea para cualquier sistema que pueda hacer uso de objetos compuestos de dicho material.

Los sistemas existentes para el reconocimiento automático de materiales trabajan únicamente sobre la información de color suministrada por una cámara estándar. En teoría, la utilización adicional de una cámara de profundidad permitiría una caracterización más fiel del material, incluyendo en el análisis factores como un rango de espectro de luz más amplio (incluyendo información en la banda infrarroja). También ofrecería la capacidad de diseñar sistemas más robustos a cambios en las condiciones de iluminación, si se ignora la información de color. Pese a las ventajas aparentes que pueda suponer idealmente el uso de las imágenes en profundidad, este campo está inexplorado en el estado del arte actual.

La motivación principal de este trabajo fin de grado es la de diseñar un prototipo de un sistema de reconocimiento automático de materiales que intente sacar el máximo partido de la información de profundidad y de la banda de infrarrojos.

1.2. Objetivos

El objetivo de este trabajo es desarrollar y diseñar dos grandes bloques que se complementen mutuamente para la generación de un prototipo capaz de distinguir e identificar automáticamente unos materiales de otros:

1. Generar una nueva base de datos constituida por una serie de materiales diferentes. Esta base de datos constará de tres tipos diferentes de imágenes obtenidas todas ellas del sensor Microsoft Kinect for Windows [4, 1, 5].

   a) Imágenes en color formato RGB

   b) Imágenes de profundidad

   c) Imágenes de infrarrojos

2. Desarrollar un algoritmo que dada una base de datos de entrada con imágenes de materiales sea capaz de generar un modelo y permita predecir nuevas muestras de forma automática.

1.3. Estructura de la memoria

La memoria del proyecto se divide en los siguientes capítulos:

- Capitulo 1. Introducción.

- Capitulo 2. Estado del arte.
- Capitulo **??**. Creación de la base de datos.
- Capitulo 4. Sistema propuesto.
- Capitulo 5. Resultados experimentales.
- Capitulo 7. Conclusiones y trabajo futuro.
- Anexos y referencias..

Capítulo 2

Estado del arte

2.1. Introducción al reconocimiento automático de materiales

El reconocimiento de materiales como se ha explicado en la sección 1.1 es algo en lo que se lleva trabajando en los últimos años debido a su posible potencial de aplicación en la vida real. Si entendemos el reconocimiento de materiales como un sistema de reconocimiento genérico, podemos representar su flujo de operaciones según el análisis clásico que se propone en la figura 2.1.

Este sistema está compuesto de cuatro fases principales:

- Selección de fuente o captura de datos: Será la primera decisión del sistema. El sistema permitirá elegir entre una cámara y una base de datos previamente capturada. Si se utiliza la cámara se pueden generar imágenes para ser procesadas a continuación de la captura o, por lo contrario, se podrá usar la información

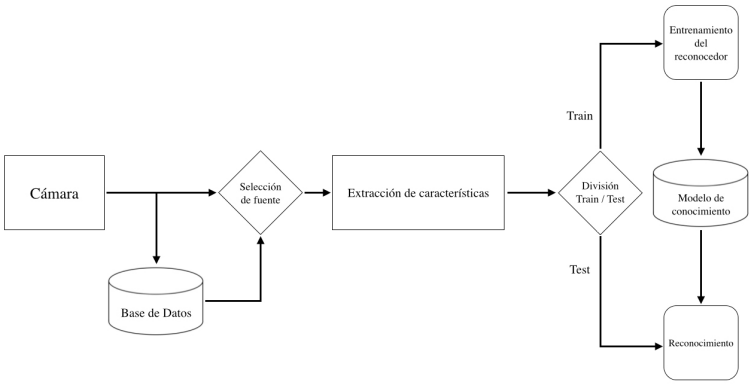


Figura 2.1: Sistema genérico de reconocimiento de materiales

capturada para generar una base de datos.

- Extracción de características: En este módulo se realiza la descripción de los datos de entrada. El objetivo del módulo será el de caracterizar los datos de entrada para poder obtener descripciones robustas y características de los materiales analizados. Este módulo variará en cada sistema implementado.
- Generación del modelo de conocimiento: En este módulo se genera un modelo de conocimiento partiendo de unos datos de entrenamiento previamente descritos mediante el extractor de características. Este modelo servirá para predecir muestras futuras.
- Testeo del modelo de conocimiento: Una vez entrenado el modelo de conocimiento podemos contrastar su rendimiento con un conjunto de muestras de test. Este módulo obtendrá por tanto, una clasificación para cada muestra de entrada, generando un porcentaje de acierto y error que podremos usar para medir cuantitativamente el rendimiento del sistema completo.

Las siguientes secciones analizan el estado del arte actual y los métodos que pueden ser potencialmente utilizados para el reconocimiento de materiales en cada fase del sistema de identificación genérico. La sección 2.2 analiza el proceso de captura de datos, haciendo especial hincapié en el uso del sensor Kinect. La sección 2.3 enumera las aproximaciones existentes y potencialmente utilizables para la caracterización de materiales. La sección 2.4 estudia los sistemas de entrenamiento y test utilizados para generar y utilizar los modelos de conocimiento. Finalmente la sección 2.5 describe las bases de datos existentes.

2.2. Captura de datos

Como se introdujo anteriormente el primer paso es la captura de imágenes para un uso del sistema en tiempo real o la generación de una base de datos. La identificación de materiales se puede llevar a cabo de diversas formas como hemos visto, la más básica utilizará simplemente imágenes en color para tratar de caracterizar una superficie y así poder reconocer un tipo de material. Sin embargo, en nuestra opinión, el reconocimiento de materiales puede verse beneficiado del uso de imágenes de profundidad e infrarrojos puesto que estas tecnologías permiten tener un detalle más preciso de una superficie. A lo largo de este trabajo se ha utilizado el sensor de Microsoft Kinect. Esta cámara es un producto habitual a día de hoy en la sociedad al tratarse de un accesorio para una videoconsola por lo que su obtención y compra es sencilla y barata, convirtiéndola en una elección óptima para el desarrollo del trabajo.

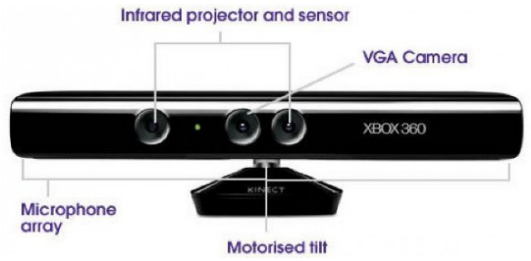


Figura 2.2: Disposición de sensores en Kinect v1

2.2.1. Microsoft Kinect v1

Esta es la primera versión que Microsoft puso a la venta del sensor en el mes de noviembre de 2010 [4]. Se trata de un dispositivo que integra varias características fundamentales y que puede ser usado tanto en la consola XBOX 360 como en un ordenador [6]:

1. Cámara VGA

   Es la cámara principal de la cámara. Se trata de una cámara RGB que obtiene las tres componentes de color. Este sensor es capaz de realizar grabaciones de 30 fps a una resolución de 640 × 480 pixeles con 8 bits por pixel.

2. Cámara y receptor de infrarrojos

   La cámara se encarga de enviar luces infrarrojas hacia la escena mientras que el receptor capta las ondas reflejadas por todo lo que se sitúe delante de la cámara. El procesador interno de Kinect genera las denominadas imágenes de profundidad. Estas imágenes son frames monocromos con una resolución máxima de 640 × 480 pixeles con 11 bit de precisión. Hay que añadir que la funcionalidad de la cámara en la captación de las imágenes de profundidad es limitada, pudiendo únicamente captar las imágenes a una distancia mínima de 1.2 m y hasta un máximo de 3.5 m.

3. Multi-Array de micrófonos

   Por ultimo Kinect también lleva incorporado un array de micrófonos que nos van a permitir funcionalidades básicas como la localización de fuentes acústicas mediante el uso de retardos como la supresión de sonido ambiente durante las grabaciones que pueda realizar la cámara.

En el año 2012 Microsoft lanzó una actualización de la cámara en la que se incluían ciertas mejoras que mejoraban su funcionamiento. Entre estas mejoras estaba la implantación del *Near Mode* que permitía usar el sensor de profundidad en el rango

de 40 cm como mínimo a 3.5 m como máximo, lo que supuso una mejora en la captación de imágenes de profundidad ya que permitía obtenerlas a una menor separación del entorno grabado.

Igualmente introdujeron una nueva aplicación: seguimiento de esqueleto que permitía realizar un tracking de un cuerpo de manera sencilla.

Y por último mejoraron la sincronización entre las imágenes en color y profundidad permitiendo un mapeo completo entre la profundidad y el color usando la nueva versión del *Software Development Kit*. La aparición de esta nueva versión del sistema es de especial relevancia, puesto que es la que permite que la captación de estas imágenes se produzca de forma completamente alineada al obtener el color en función de la profundidad. Alineamiento (o calibración) ésta que permite el análisis de ambas informaciones de manera sincronizada tanto espacial como temporalmente.

2.2.2. Microsoft Kinect v2

La segunda versión de la cámara se empezó a comercializar a mediados del 2014. Fue una actualización completa de la primera versión donde además de cambiarse el sensor, incluía diversas y significativas mejoras con respecto a su predecesora [1].

Esta nueva actualización permitía también el uso de la cámara tanto en el sistema de XBOX One como en un PC de sobremesa (usando un adaptador cuya compra es necesario realizar aparte) haciendo uso de la ultima versión del SDK de Microsoft [5]. Las características principales de esta nueva cámara son las siguientes:

1. Camara RGB

   Se actualiza el sensor de la cámara a color actualizando los formatos a RGB y YUV e incluyendo una resolución máxima de 1080x1920p a una resolución temporal de 15 fps.

2. Camara de profundidad

   El sensor de profundidad sufre una considerable mejora actualizando su resolución a 512x424p y sobre todo aumentando el rango de funcionamiento desde los 0.4m a los 8m lo que supone un aumento significativo en el rango físico de captación de imágenes con respecto a la primera versión de Kinect.

3. Multi-Array de micrófonos

   Los micrófonos no tienen prácticamente actualización y simplemente siguen realizando las mismas funcionalidades que en la versión 1.

Figura 2.3: Diseño físico de Kinect v2

2.2.3. Principales diferencias

Si comparamos de forma más directa los dos sensores podemos dividir la caracterización de las diferencias en dos bloques, el primero el rango de visión de ambos sistemas y el segundo, las diferencias en el flujo de datos que podemos obtener de cada una de las cámaras.

RANGO DE VISIÓN		
Característica	Kinect XBOX 360	Kinect XBOX One
Campo de visión horizontal	57º	70º
Campo de vision vertical	43º	60º
Rango de inclinación física	+-27º	+-27º
Rango de profundidad del sensor	1,2 - 3,5 metros	Default Mode
		0 – 0,8 metros, fuera de rango. 0,8 – 4,5 metros, parámetros normales. 4,5 – 8 metros, se recoge información pero no es óptima. > 8 metros, fuera de rango.
		Near Mode
		0 – 0,4 metros, fuera de rango. 0,4 – 3 metros, parámetros normales (la mejor calidad se encuentra a los 2 metros). 3 – 8 metros, se recoge información pero no es óptima. 8 metros, fuera de rango.

Tabla 2.1: Diferencias rango de visión entre Kinect v1 y v2

Aunque el *Near Mode* se incluyó ya en la revisión de Kinect v1, no ha sido hasta la comercialización de Kinect v2 cuando se le ha dado un verdadero uso. Si nos fijamos en la figura 2.4 podemos comprobar gráficamente lo que supone el uso tanto del *Default Mode* (modo de uso habitual de la cámara) como el *Near Mode* .

Como podemos observar en la imagen las medidas están realizadas sobre los rangos de visión de Kinect v2 (0,4 - 8m) y podemos visualizar el efecto que produce el uso del *Near Mode* [7]. Este modo nos va a permitir obtener valores de profundidad óptimos

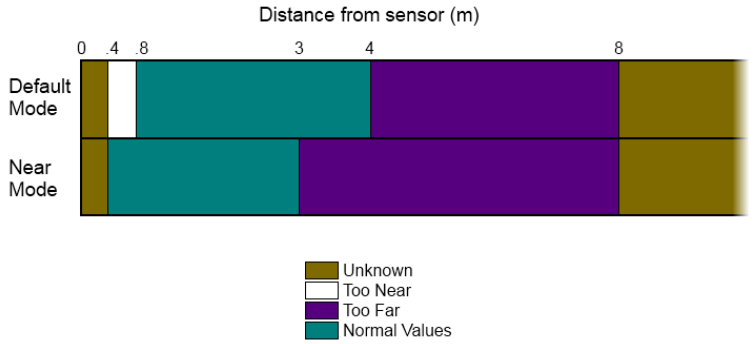


Figura 2.4: Diferencias Default y Near Mode en Kinect v2

en el rango de 0,4 a 3 metros frente al rango de 0,8 a 4 metros del modo por defecto. Esto va a permitir la captura de imágenes correctas de profundidad a una distancia más cercana al sensor lo que va a suponer una mejora significativa en el proceso de captación. Por lo tanto la combinación de ambos modos tendrá como resultado una mejor captura general de las imágenes de profundidad.

Si pasamos ahora a comparar las diferencias entre los diferentes flujos de datos que podemos obtener de ambas versiones obtenemos la siguiente tabla de la figura 2.2.

DATA STREAM			
Tipo de flujo	Caracteristica	Kinect XBOX 360	Kinect XBOX One
Imágenes en color	Resolución	640 × 480p VGA	640 × 480 RGB 1280 x 960 RGB 640 x 480 Raw YUV 1080 x 1920p RGB
	Precisión	32 bits	32-bit de color
	Frame Rate	30 fps	30fps 12fps 15fps 15fps
Imágenes en profundidad	Resolución	320 × 240p	80 x 60p 320 × 240p 512 × 424p
	Precisión	16 bit	16 bit
	Frame Rate	30fps	30fps
Audio	Frecuencia de muestreo	16 kHz	16 kHz
	Precision	16 bit	16 bit 64 bits

Tabla 2.2: Diferencias flujos de datos entre Kinect v1 y v2

Todas estas mejoras descritas en las tablas 2.1 y 2.2 suponen un aumento de

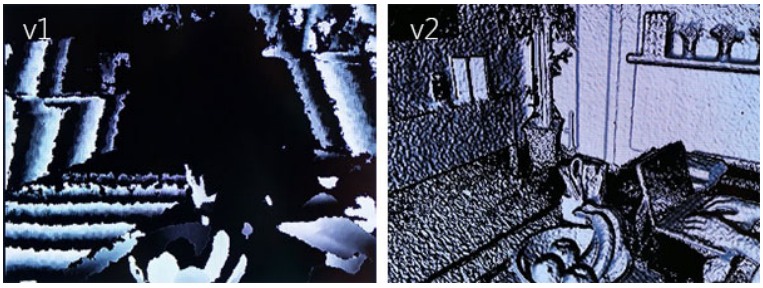


Figura 2.5: Diferencias entre imágenes de profundidad entre Kinect v1 y Kinect v2. Imágenes tomadas de [1]

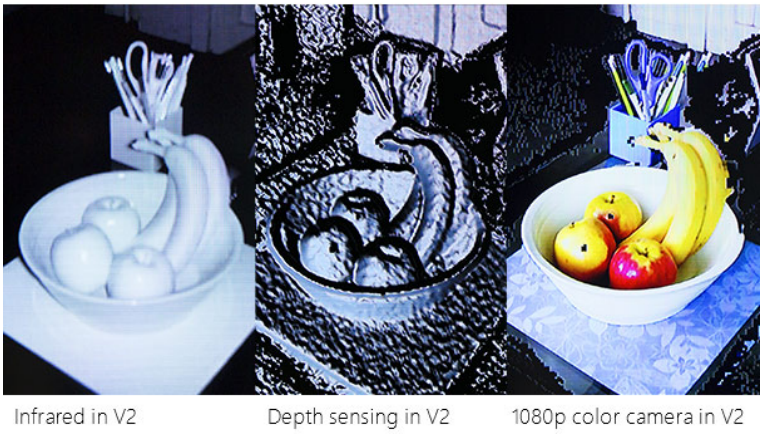


Figura 2.6: Diferentes imágenes captadas con Kinect v2. Imágenes tomadas de [1]

la calidad general de las imágenes obtenidas con la cámara. Si nos fijamos en la figura 2.5 podemos observar como repercute directamente la precisión de la cámara de profundidad en el nivel de detalle final captado de una escena.

Y del mismo modo lo podemos comprobar con las imágenes de infrarrojos y de color de la figura 2.6

Este aumento de precisión en la captura puede ser un factor determinante si finalmente tenemos que utilizar muestras de objetos de un tamaño reducido a una distancia lejana para poder captar diferencias de profundidad correctamente.

2.3. Caracterización de los materiales

La caracterización de materiales es el segundo bloque en nuestro sistema genérico y en concreto la descripción de su superficie. En nuestra opinión, no existe aún un descriptor perfecto, que se adapte a todas las situaciones tanto de iluminación como de posición de la cámara respecto a los materiales capturados. Sin embargo, podemos

encontrar diferentes tipos de descriptores cada uno con sus ventajas e inconvenientes. Actualmente existen algunos que se han propuesto y utilizado en sistemas de reconocimiento y que se detallaran a continuación. Adicionalmente, incluimos algunos descriptores que si bien no han sido usados para el reconocimiento de materiales, presentan una gran capacidad de uso potencial. Algunos de éstos serán utilizados durante este trabajo fin de grado, otros se proponen como parte de un potencial trabajo futuro.

2.3.1. Descriptores actualmente utilizados para reconocimiento de materiales

Descriptores implícitos del material

En [8] se proponen una serie de descriptores para abordar el problema del reconocimiento de materiales analizando únicamente información de color. La estrategia propuesta es la de usar un conjunto de características de medio y bajo nivel que capturen información de varias categorías de características.

1. Color y textura

   El color es un atributo importante que se puede usar para la identificación de materiales puesto que en algunos casos suele ser bastante discriminativo aunque nunca infalible. En esta aproximación se incluyen dos descriptores fundamentales para obtener información del color y la textura como son Jet (también denominados filtros de Gabor) [9] y SIFT [10].

2. Microtextura

   Para caracterizar la microtextura se propone usar una ampliación de SIFT y Jet. Esta ampliación se basa en la aplicación de los descriptores sobre una imagen denominada residual. Esta imagen siguiendo la idea de [11] se calcula mediante la diferencia entre la imagen original y la imagen original filtrada bilateralmente [12]. Estos nuevos descriptores serán denominados micro-jet y micro-SIFT.

3. Contorno de forma

   Aunque los materiales pueden tener prácticamente cualquier forma dependiendo de su origen, suelen predominar más unas formas que otras. De esta forma se propone extraer los contornos de una forma en una imagen mediante un mapa de bordes. Esto se va a realizar mediante el detector de Canny [13] en la imagen inicial seguido de la eliminación de los contornos de menor longitud y de la obtención de un mapa de contornos. Para caracterizar las variaciones a lo largo

de este mapa se propone medir la curvatura de los contornos del mapa a lo largo de tres escalas diferentes.

4. Reflectancia

   Normalmente las propiedades de reflectancia de un material, como pueden ser la trasparencia o el brillo, se suelen reflejar en variaciones características de la intensidad en los bordes de una imagen. Para medir estos cambios se propone utilizar los histogramas de gradientes orientados (HoG) a lo largo de la dirección normal de los contorno [14]. De esta forma se obtienen los HoGs a lo largo de un contorno determinado. A esta característica se le denomina "edge-slice".

Local Orientation Adaptive Descriptor (LOAD)

En [15], un artículo aún en proceso de revisión, se propone un nuevo descriptor para el reconocimiento de materiales dada una imagen en color. Su objetivo es diseñar un método de caracterización capaz de discriminar texturas teniendo ademas dos propiedades fundamentales:

1. Discriminación regional de texturas

   Muchos descriptores entre los que se incluyen los comentados en la sección 2.3.2 están diseñados para la búsqueda de estructuras en imágenes. Un ejemplo de ello son todos los utilizados para el reconocimiento de personas o objetos en tramas de video [16, 17]. Su desempeño en estos ámbitos es notable, sin embargo no lo es tanto en la descripción de texturas en imágenes en color. El diseño del descriptor LOAD se centra en capturar información discriminativa acerca de la textura regional utilizando sólo información de la variabilidad de luminancia alrededor de un punto.

2. Robusto a transformación de la imagen

   Como se ha comentado anteriormente las imágenes naturales son muy susceptibles de contener transformaciones geométricas de los materiales capturados respecto a las vistas entrenadas. Estas transformaciones incluyen: rotaciones, cambios de escala y de iluminación. El descriptor que se propone debe ser invariante a estas transformaciones.

En el transcurso del trabajo fin de grado se planteó el uso de LOAD como descriptor alternativo a los finalmente utilizados. Finalmente esta opción fue desechada por restricciones temporales. Por este motivo, se incluye una descripción detallada del algoritmo en el Anexo 7.1.

2.3.2. Descripciones de potencial utilidad para el reconocimiento de materiales

Bidirectional Reflectance Distribution Function (BRDF)

Todos los descriptores comentados anteriormente basan su desarrollo en caracterizar una superficie en función de la luz que incide sobre ellos, en particular, de la luz que incide sobre ellos y ha sido capturada por la cámara. Sin embargo existe uno con el que podemos tener la mejor aproximación acerca de estas características de una superficie. Este descriptor se conoce como *Bidirectional Reflectance Distribution Function* (BRDF).

BDRF [18] se trata de una función que nos dará las propiedades de reflectancia de una superficie, es decir, nos proporcionará información acerca de como la luz se refleja en una superficie opaca en función de cuatro variables reales.

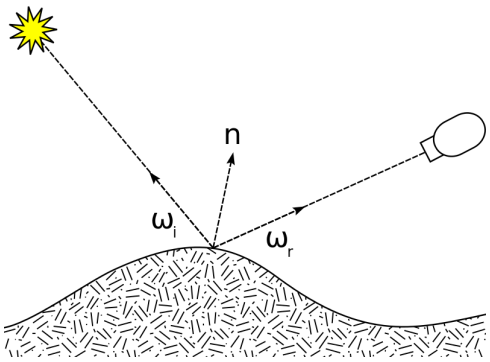


Figura 2.7: Diagrama con los vectores usados por BRDF

Como podemos observar en la figura 2.7 la función toma una dirección de la luz incidente ω_i y una luz reflejada ωr tomada con respecto a un sistema de coordenadas en la que la normal de la superficie n se sitúa a lo largo del eje z y devuelve el ratio de reflectancia existente a lo largo de ωr con respecto a la irradiancia incidente en la superficie de dirección ω_i. Cada dirección ω esta parametrizada por un angulo azimutal ϕ y cenital θ. Por lo tanto la función tiene 4 variables.

La propiedad de reflectancia se podrá correlar habitualmente con las categorías de materiales, siendo por tanto características de un tipo u otro de material. Sin embargo, podremos encontrar el caso de que materiales que se sitúen en diferentes categorías, tengan propiedades de reflectancia similares y que por tanto la BRDF nos sea decisiva para una identificación correcta.

Este descriptor pese a esto se postula como la opción perfecta para describir un material puesto que nos va a permitir describir con una precisión alta su superficie.

Sin embargo, el mayor inconveniente de esta función es que es muy sensible a cambios de iluminación, lo que nos obliga a tener un entorno de identificación de materiales muy controlado, tanto en la cantidad de iluminación como en el ángulo de incidencia de la misma en el objeto en la grabación de una base de datos que posteriormente se caracterice mediante el uso de esta función. Además, su cálculo es realmente complicado dado que es difícil obtener los ángulos de iluminación y captura de una escena y aislar estos factores del cálculo de la normal n, además es un proceso costoso y su único uso actualmente se ha dado mediante aproximaciones a la función. Estas aproximaciones pueden ser tanto en dos como en tres dimensiones y se detallan a continuación:

Aproximaciones 2D a BRDF

Durante esta sección se van a comentar una serie de aproximaciones a la reflectancia de una superficie y por tanto a la BDRF de un material. Estos descriptores van a ser útiles para el sistema dado que van a caracterizar la superficie de un material de una forma más simple y eficiente que la BDRF pero sin embargo pueden ser capaces de describir con suficiente precisión las diferentes características de reflectancia de un material para poder clasificarlo más adelante.

Histogram of Oriented Gradients (HoG)

HoG es un descriptor de características usado principalmente para la detección de objetos en secuencias de imágenes [16, 17]. Este algoritmo basa su funcionamiento en que la forma y apariencia de un objeto en una imagen puede ser descrita mediante un conjunto de gradientes de intensidades organizados a su vez en histogramas en función de sus orientaciones. Estos histogramas describen la distribución de las intensidades de los gradientes locales o la dirección de los bordes.

El proceso de cálculo del HoG se resume a continuación, siguiendo [14].

La imagen $I(x, y)$ se divide en regiones denominadas celdas que serán contiguas y que ocuparan toda la imagen. El numero de celdas en las que se divide la imagen será un parámetro de configuración del algoritmo al que denominaremos N_s.

La versión básica de HOG estima la derivada de gradiente de una imagen mediante su convolución con las siguientes máscaras:

$$F_x = [-1, 0, 1] \quad F_y = [1, 0, -1] \tag{2.1}$$

Con estas mascaras podremos calcular tanto el gradiente en la dirección horizontal como el gradiente en la dirección vertical (denotados como I_x e I_y respectivamente)

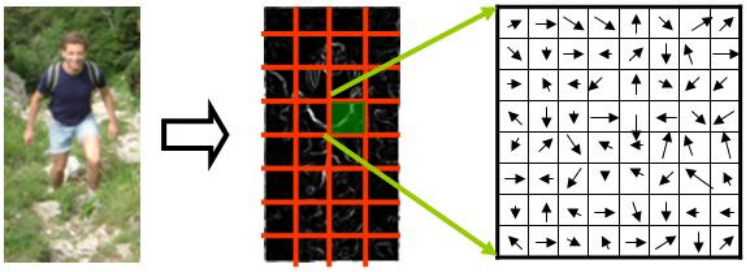


Figura 2.8: Ejemplo HoG

mediante el filtrado de la imagen original con ambas mascaras:

$$I_x(x,y) = I(x,y) * F_x \tag{2.2}$$

$$I_y(x,y) = I(x,y) * F_y \tag{2.3}$$

Una vez que ambas imágenes hayan sido calculadas, la magnitud $M(x,y)$ y la orientación $\theta(x,y)$ del gradiente se obtienen como:

$$\theta(x,y) = atan\left(\frac{I_y(x,y)}{I_x(x,y)}\right) \tag{2.4}$$

$$M(x,y) = \sqrt{I_y(x,y)^2 + I_x(x,y)^2} \tag{2.5}$$

En este punto del algoritmo tendremos para cada pixel de la imagen información de la magnitud y el ángulo del gradiente. Para cada una de las celdas descritas anteriormente se construye un vector de características formado por la concatenación de los histogramas de ángulos y magnitudes de cada pixel.

La figura 2.8 muestra un ejemplo de generación del HoG. La imagen se divide en celdas y de cada una de ellas se obtiene un valor de magnitud (representado en la figura 2.8 como la longitud de la flecha) y una orientación (representada por la dirección y el sentido de la misma).

El HoG se trata por tanto de un descriptor bastante discriminante en cuanto al reconocimiento de formas en una imagen, sin embargo tiene un problema, y es que no es invariante a rotación. Esto va a suponer una limitación enorme si se quiere utilizar para el reconocimiento de materiales puesto que será prácticamente imposible encontrar dos extractos de materiales idénticamente orientados.

Sin embargo en [3] se proponen dos extensiones de HoG que pueden superar la limitación de la invarianza a la rotación previamente comentada. Estos nuevos métodos van a estar basados en filtros Gaussianos derivativos (GDF-HoG) y en la matriz Hessiana (EigHess- HoG). En vez de utilizar los métodos convencionales explicados

para el cálculo del algoritmo HoG, usaran las derivadas parciales de segundo orden tanto de los filtros Gaussianos como de la matriz Hessiana que proporcionan mayor estabilidad para calcular intensidad y variación de textura en una imagen de superficie. A continuación se explican en detalle el cálculo de ambos algoritmos.

Gaussian derivative filters - Histogram of Oriented Gradients (GDF-HoG) Se trata del primer método expuesto en [3] y que mediante el uso de las primeras derivativas de la imagen mediante la función Gaussiana permite suministrar al proceso invarianza a la rotación. La rotación al igual que el escalado o la traslación son operaciones de transformación lineal, es decir, aunque las coordenadas de la imagen cambien las transformaciones comentadas mantienen la forma de la imagen intacta. La función Gaussiana se trata de una función continua y lineal, por lo que su uso en el cálculo de la primera y segunda derivativas permiten suministrar invarianza a rotación como se demuestra en [19]. Además el cálculo del gradiente mediante la función Gaussiana y la convolución bidimensional permite la extracción de forma más fiable de la información de microtextura de la imagen, mejorando por lo tanto, la descripción de la imagen.

Definimos la función Gaussiana de la siguiente forma:

$$G(x,y) = e\frac{-(x^2+y^2)}{\sigma^2} \tag{2.6}$$

Para el calculo del algoritmo GDF-HOG esta vez se usan $G_x(x,y)$ y $G_y(x,y)$, las primeras derivativas direccionales de la Gaussiana, para el cálculo de las derivadas parciales de la imagen. Pueden definirse:

$$G_x(x,y) = \frac{\partial G(x,y)}{x} \tag{2.7}$$

$$G_y(x,y) = \frac{\partial G(x,y)}{y} \tag{2.8}$$

La información del gradiente se calcula mediante las ecuaciones 2.2 y 2.3 siendo las mascaras básicas F_x y F_y sustituidas por $G_x(x,y)$ y $G_y(x,y)$ y obteniendo así nuevas versiones de $I_x(x,y)$ e $I_y(x,y)$.

Podremos de esta forma obtener la magnitud del gradiente de la imagen analizada $I(x,y)$ como:

$$M(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2} \tag{2.9}$$

El ángulo del gradiente se calcula mediante la ecuación:

$$\theta(x,y) = arctan\left(\frac{I_y(x,y)}{I_x(x,y)}\right) \tag{2.10}$$

De esta manera obtendremos al igual que en en el cálculo de la versión básica de HoG una magnitud o intensidad y un ángulo para cada pixel de la imagen. De la misma forma obtenemos un histograma por celda y concatenando las celdas conseguiremos el descriptor completo de la imagen GDF-HOG.

Eigenvalues of the Hessian matrix - Histogram of Oriented Gradients (EigHess- HOG)

El segundo método descrito en vez de utilizar los filtros derivativos Gaussianos utiliza la matriz Hessiana [20] para calcular los autovalores de la imagen de superficie. La matriz Hessiana de una imagen esta definida como una matriz de derivadas parciales de segundo orden de esa imagen en escala de grises $I(x,y)$. Esta matriz $H(x,y)$ tiene autovalores reales al tratarse de una matriz de valores reales.

Si definimos $G_{xx}(x,y)$, $G_{yy}(x,y)$, $G_{xy}(x,y)$ y $G_{yx}(x,y)$ como los filtros derivativos Gaussianos de segundo orden de la imagen a lo largo de x, y, xy y yx respectivamente podemos obtener la matriz Hessiana $H_\sigma(x,y)$ de un punto de una imagen en escala de grises $I(x,y)$ para una escala σ de la manera que sigue:

$$H_\sigma(x,y) = \begin{bmatrix} D_{xx}(x,y) & D_{xy}(x,y) \\ D_{yx}(x,y) & D_{yy}(x,y) \end{bmatrix} = \begin{bmatrix} I(x,y) * G_{xx}(x,y) & I(x,y) * G_{xy}(x,y) \\ I(x,y) * G_{yx}(x,y) & I(x,y) * G_{yy}(x,y) \end{bmatrix} \tag{2.11}$$

Donde $D_{xx}(x,y)$, $D_{yy}(x,y)$, $D_{xy}(x,y)$ y $D_{yx}(x,y)$ serán por tanto las derivadas de segundo orden de la imagen a lo largo de las direcciones x, y, xy y yx respectivamente.

La matriz Hessiana en general contiene más información que la computación de gradientes. Las operaciones de primer orden suelen ser insuficientes para describir el comportamiento de una función no lineal como si lo hacen las de operaciones de segundo orden [20].

La principal idea sobre la que se basa este método es la de extraer las direcciones y curvaturas principales de una imagen de superficie. Denominamos a los autovalores del Hessiano curvaturas primarias e invariantes a rotación y se pueden definir de la siguiente forma [21]:

$$\lambda_1(x,y) = \sqrt{\frac{\left(I(x,y) * G_{xx}(x,y) - I(x,y) * G_{yy}(x,y)\right)^2}{4} + \left(I(x,y) * G_{xy}(x,y)\right)^2} + K \tag{2.12}$$

$$\lambda_2(x,y) = -\sqrt{\frac{\left(I(x,y) * G_{xx}(x,y) - I(x,y) * G_{yy}(x,y)\right)^2}{4} + \left(I(x,y) * G_{xy}(x,y)\right)^2} + K \tag{2.13}$$

, con:

$$K = \frac{I(x,y) * G_{xx}(x,y) - I(x,y) * G_{yy}(x,y)}{2} \tag{2.14}$$

Donde $\lambda_1(x,y)$ y $\lambda_2(x,y)$ son los autovalores de la matriz Hessiana. Esto nos va a permitir obtener las principales direcciones en la imagen. El resultado gráfico del calculo de los autovalores puede observarse en la figura 2.9.

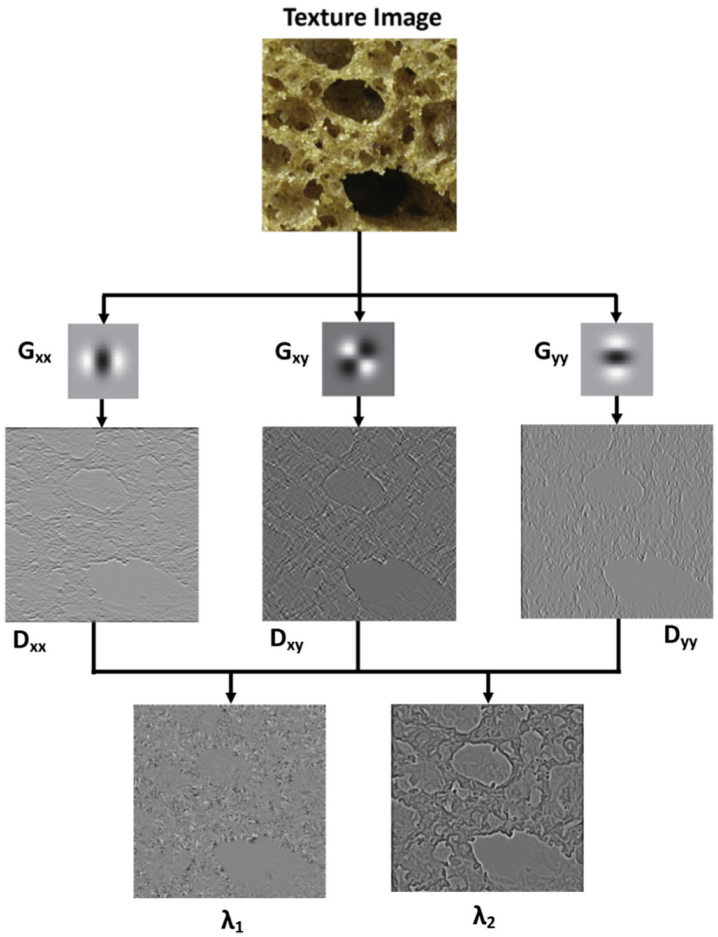


Figura 2.9: Procedimiento para calcular los autovalores de la matriz Hessiana y resultados obtenidos. La imagen esta sacada de la base de datos [2] y el procedimiento de [3]

Finalmente y una vez obtenidos $\lambda_1(x,y)$ y $\lambda_2(x,y)$ podemos utilizar sus valores para calcular la magnitud del gradiente y su orientación sustituyendo en este caso al cálculo convencional de la magnitud y orientación del gradiente. En particular:

$$M(x,y) = \sqrt{(\lambda_1(x,y))^2 + (\lambda_2(x,y))^2} \tag{2.15}$$

$$\theta(x,y) = arctan\left(\frac{\lambda_2(x,y)}{\lambda_1(x,y)}\right) \tag{2.16}$$

De nuevo volvemos a realizar los histogramas comentados en métodos anteriores y su concatenación obteniendo finalmente el vector de características de la imagen.

El trabajo fin de grado que se presenta hará uso de los descriptores EigHess- HOG y GDF-HoG para el reconocimiento de materiales.

Aproximaciones 3D

Ademas de las aproximaciones comentadas en la sección 2.3.2 existen descriptores más complejos que tratan de estimar de una forma más precisa la BDRF de una superficie dada su representación volumétrica. Estos descriptores permitirían caracterizar de forma más precisa las reflexiones de luz que se producen en los materiales así como de los ángulos en los que se producen.

Signature of Histograms of OrienTations (SHOT)

En [22] se estudia el caso del alineamiento de superficies usando características 3D. Para ello utilizan descriptores de una dimensión mayor pasando de aquellos que solo caracterizaban en dos dimensiones a los que utilizan información de las tres dimensiones completas. Para ello en [22] se hace una diferencia entre dos grupos diferentes de descriptores.

1. Descriptores *Signature*

   Son aquellos descriptores que caracterizan de forma tridimensional los pixeles del vecindario de un punto dado definiendo primero una referencia invariante local y codificando de acuerdo a estas coordenadas locales una o más medidas geométricas calculadas individualmente.

2. Descriptores basados en histogramas

   Estos descriptores caracterizan el vecindario acumulando medidas topológicas y geométricas locales, como por ejemplo cuentas de puntos, o áreas de triángulos entre otros, en histogramas de acuerdo a un dominio especifico.

En resumen, los descriptores *Signatur*e son altamente descriptivos gracias al uso de información localizada mientras que los histogramas mantienen un equilibrio entre poder de descripción y robustez comprimiendo estructuras geométricas en bins.

En [22] se propone el uso de un descriptor 3D que parte de una combinación tanto de descriptores *Signatur*e como de descriptores basados en histogramas. Este descriptor codifica histogramas de entidades de primer orden básicas que son más representativas de la estructura local de una superficie que lo que lo pueden llegar a ser las coordenadas tridimensionales. El uso de los histogramas tiene la idea básica de eliminar el posible efecto de errores y así conseguir robustez frente a ruido.

La idea del descriptor es la de introducir información geométrica sobre los puntos en el vecindario del vecindario imitando de ese modo un descriptor *Signatur*e. Para realizar este proceso se propone calcular una serie de histogramas locales sobre volúmenes tridimensionales definidos mediante una malla alrededor del vecindario para después concatenar todos los histogramas locales y formar así el descriptor completo.

El calculo de cada histograma local se realiza de forma parecida a un descriptor HoG es decir, acumulando puntos del histograma en función del ángulo θ_i que forman la dirección normal de un punto correspondiente a la malla n_{vi} y la normal en el punto característico n_u.

El uso de este descriptor se planteó como alternativa a los finalmente utilizados en el transcurso del trabajo fin de grado que se presenta, de ahí su inclusión en este apartado de la memoria y como parte del potencial trabajo futuro.

2.4. Generación del modelo de conocimiento

2.4.1. LDA

Se trata de un modelo generativo propuesto en [23] que va a permitir explicar conjuntos de observaciones a partir de grupos no supervisados. Existe ademas una modificación sobre este algoritmo denominada aLDA que nos va a permitir solucionar un problema común en la generación de modelos. Habitualmente el uso de todas las características e incluso simplemente el uso de una elevada cantidad de muestras de entrenamiento puede provocar un sobreentrenamiento del sistema empeorando de esta forma el rendimiento en la fase de test si el modelo está sobre-ajustado a los datos de entrenamiento. Para evitarlo se propone en [8] el uso de aLDA. La idea principal es elegir la mejor característica que maximice el ratio de reconocimiento en el grupo de test. De esta forma se van añadiendo características en la generación de un modelo LDA hasta que se llega al punto en el que si se añaden más el rendimiento decrece.

Ni LDA ni aLDA serán utilizados en el desarrollo de este trabajo fin de grado, sin

embargo han sido utilizados con éxito en sistemas anteriores para el reconocimiento de materiales [23].

2.4.2. Support Vectors Machine (SVM)

Las Maquinas de Vector Soporte son un método de clasificación de datos formado por una serie de algoritmos de aprendizaje supervisado. Habitualmente en los problemas de regresión y clasificación se tiene acceso a un conjunto de muestras. El proceso de clasificación tiene como norma la separación de este conjunto en dos subconjuntos denominados conjunto de entrenamiento y conjunto de test. Cada muestra del conjunto de entrenamiento va a tener un valor de clasificación, como por ejemplo una etiqueta de clases, y una serie de atributos como pueden ser las características obtenidas mediante una serie de descriptores.

El objetivo final de SVM es producir un modelo basado en este conjunto de entrenamiento que prediga los valores de clasificación del conjunto de test únicamente proporcionándole sus atributos.

La idea general es que los vectores de entrenamiento son mapeados en un espacio dimensional mayor, incluso infinito, y son separados mediante un hiperplano o conjunto de hiperplanos por un espacio lo más amplio posible. La separación entre clases óptima es la característica fundamental de SVM puesto que el objetivo es que el hiperplano que se defina como la solución idónea tenga la máxima distancia con los puntos que estén más cerca de él, denominaremos a esta distancia margen.

Podemos observar este concepto en la figura 2.10 representada mediante el modelo más básico de SVM (2D).

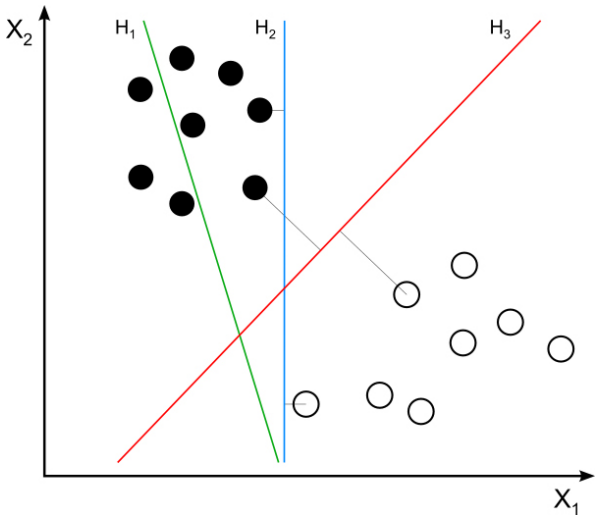


Figura 2.10: Ejemplo en 2D de SVM

Como podemos observar tenemos dos clases diferentes en los datos de entrenamiento que forman dos conjuntos. No existe un único hiperplano (en este caso un hiperplano en 2D es una linea) que sea solución única del problema. Sin embargo existen soluciones que idealmente nos proporcionaran un mejor resultado de clasificación más adelante.

Como se ha comentado anteriormente el objetivo de SVM es encontrar el hiperplano que tenga mayor margen con respecto a los datos, puesto que aunque H2 separa correctamente las clases lo hace de una forma peor que H3 que tiene más margen con respecto a los mismos.

Desafortunadamente estos casos ideales no se dan en la realidad y en la vida real SVM debe de ser capaz de generar un modelo que pueda tratar con:

1. Clasificaciones de más de dos categorías o clases

2. Separación entre clases no lineal

3. Clases que no estén perfectamente separadas

La solución a estos problemas comentados es el uso de funciones Kernel. Esta función va a ser la responsable de proyectar nuestra información de las muestras en un espacio de características mayor permitiendo una mejor separación entre clases y reduciendo ademas el coste computacional de creación del modelo.

2.5. Bases de datos disponibles para el reconocimiento de materiales

Existen tres bases de datos con imágenes de objetos fabricados en diferentes materiales. La primera es *Flickr Materials Database* (FMD)[24].

Esta base de datos contiene las categorías más importantes de materiales presentes en la vida real (materiales fabricados, follaje, vidrio, cuero, metal, papel, plástico, piedra, agua y madera). FMD tiene 100 imágenes por categoría, es decir 1.000 imágenes en total, en las cuales se presenta una variedad de condiciones de iluminación, composiciones, colores y texturas, de el amplio rango de objetos comentado unas lineas atrás. Se trata por tanto, de una base de datos realmente completa con imágenes a color y con alta resolución que nos muestran varios objetos del mismo tipo de material en la misma imagen.

Figura 2.11: Ejemplos imágenes *Flickr Materials Database*

La segunda base de datos es *KTH-TIPS Material Database*[2]. Se trata de una base de datos de 820 imágenes de 200x200 pixeles de una serie de materiales que contienen cada uno de ellos una vista cercana de cinco objetos diferentes del mismo material (aluminio, piel de naranja, arena... etc) a diferentes escalas. En total cada material se fotografía a nueve escalas diferentes y de cada escala se toman nueve imágenes teniendo por tanto 81 imágenes para cada material. Esta base de datos se diferencia de la anterior en que esta no representa objetos enteros, si no una vista aumentada del material del que está formado el objeto fotografiado.

Figura 2.12: Ejemplos imágenes *KTH-TIPS Material Database*

La ultima base de datos es *MPI-VIPS* [2]. Se trata de una actualización de la base de datos anterior que trata de mejorar las imágenes de algunos materiales captando más detalle, y además añade nuevos materiales como son por ejemplo madera, lana o algodón. Esta nueva base de datos toma todas las fotografías a la misma escala perdiendo de esta forma información sobre la variable escala, pero sin embargo añadiendo una nueva variante como es la iluminación. Cada material consta de 27 imágenes de cada uno de los 4 tipos de iluminación, que incluyen iluminación de ambiente, iluminación artificial desde el frente, desde la derecha y desde la izquierda, formando así un conjunto de 108 imágenes de cada material.

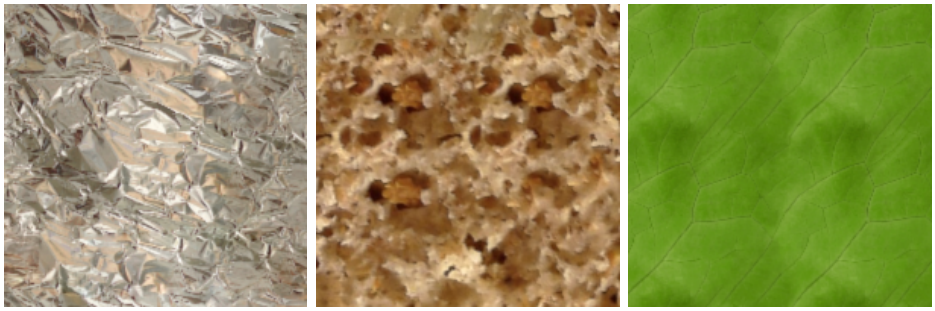


Figura 2.13: Ejemplos imágenes *MPI-VIPS*

Como se ha comentado en la sección 1 no existe ninguna base de datos actualmente disponible que haga uso tanto de imágenes de infrarrojos como de imágenes de profundidad, por lo que una de las bases del trabajo es la creación de una base de datos para el reconocimiento de materiales añadiendo información sobre este tipo de imágenes.

Capítulo 3

Creación de la base de datos

Se trata del primer objetivo del trabajo. Como se ha comentado anteriormente, no existe actualmente una base de datos que incluya los siguientes tres tipos de imágenes para la detección de materiales:

1. Imágenes en color formato RGB

2. Imágenes de profundidad

3. Imágenes de infrarrojos

Tanto las imágenes de profundidad como las de infrarrojos pueden describir de nuevas formas los materiales y por tanto pueden añadir nueva información que nos permita ser más precisos en su identificación automática.

El capítulo está organizado de la siguiente manera: En la sección 3.1 se describen los materiales utilizados y se explican los motivos por los que se han seleccionado. En la sección 3.2 se describe el entorno de grabación y en la sección 3.3 el *software* utilizado para la captura de las imágenes de los materiales seleccionados. Finalmente, en la sección 3.4 se describe el proceso de aislamiento de los materiales de su entorno.

3.1. Materiales

La primera decisión que se ha tomado en la grabación de la base de datos ha sido seleccionar del amplio abanico de materiales existentes una cantidad manejable que vamos a analizar. Los once materiales seleccionados finalmente se incluyen en la Tabla 3.1.

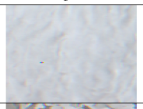
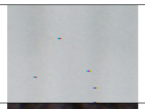
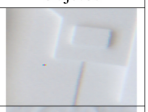
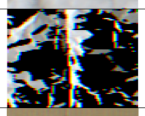
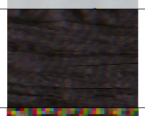
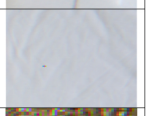
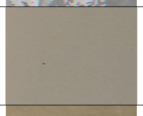
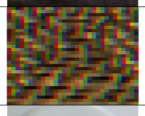
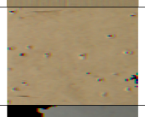
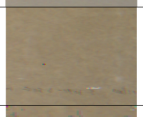
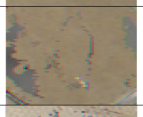
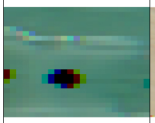
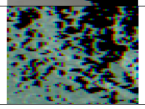
IMAGENES					
MATERIAL	Primeros Objetos	Segundos Objetos	MATERIAL	Primeros Objetos	Segundos Objetos
Algodón			Poliespan		
Aluminio			Tela		
Carton			Mimbre		
Madera			Porcelana		
Metal			Cristal		
Papel de burbujas					

Tabla 3.1: Materiales elegidos

La decisión de la selección de estos materiales viene determinada por dos motivos. El primer motivo radica en que la base de datos se ha diseñado como una mejora de las bases de datos propuestas en [2] y [24], que han servido de primera aproximación para la selección de materiales. El segundo motivo se debe a que estos materiales se pueden encontrar diariamente en la vida real, lo que va a suponer dos ventajas: los materiales elegidos son fáciles de manejar y conseguir para la grabación; y su uso incrementa la potencial utilidad del prototipo descrito en el capítulo 4.

Para la grabación de la base de datos se propuso inicialmente seleccionar once objetos diferentes cada uno fabricado en cada uno de los once materiales descritos en este punto (versión 1 de base de datos). Sin embargo el hecho de tener un conjunto de imágenes de diferentes materiales formado solo por un conjunto de objetos tiene un problema. Puede que el sistema funcione correctamente pero al estar modelando siempre los mismos once objetos el trabajo pasaría de tratarse de un problema de identificación automática de materiales a identificación automática de once objetos. Por esta razón y para solventar esta cuestión se propuso una segunda grabación de idéntico procedimiento de los once mismos materiales definidos pero que estuvieran presentes en otros objetos diferentes (versión 2 de la base de datos). De esta forma por tanto tendremos una base de datos donde cada material estará representado en dos objetos diferentes.

(a) Vista 1 (b) Vista 2

Figura 3.1: Entorno de grabación en el VPU

3.2. Entorno y equipo de grabación

La grabación se ha realizado en el laboratorio Video Processing and Understanding Lab (VPU) de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. Como se ha comentado en el capítulo introductorio 2.2 se ha utilizado una cámara Microsoft Kinect de segunda generación para la grabación de la base de datos.

El objetivo fundamental de este punto del trabajo era obtener un conjunto de imágenes captadas en diferentes condiciones de captura, iluminación y ángulo de incidencia de iluminación para disponer de una base de datos controlada con la que poder medir la invarianza a la rotación y a las condiciones de iluminación del prototipo desarrollado (capítulo 4). Con este objetivo, se ha diseñado un sistema rotatorio por el cual se puede colocar un material en un soporte e ir rotándolo de forma sucesiva para así capturar ese material desde diferentes puntos de vista. Además, para variar las condiciones de iluminación se han obtenido imágenes capturadas tanto con la luz ambiente del laboratorio como con una lampara de luz LED.

Las siguientes imágenes muestran tanto el entorno de grabación (en las figuras 3.1a y 3.1b), como el sistema rotatorio diseñado (en la figura 3.2).

Como podemos observar en la figura 3.1, la grabación se ha realizado utilizando un soporte vertical para la sujeción de la cámara de tal forma que esté en un ángulo lo más cercano posible a la perpendicularidad con la mesa. Cuanto más cerca esté el angulo de los 90^{o} , menos variaciones tendremos en las imágenes de profundidad de objetos planos y más sencillo será el proceso de aislamiento del material descrito en la sección 3.4. También puede observarse la lámpara de luz LED usada para iluminar la escena con un foco de luz adicional a la luz ambiente.

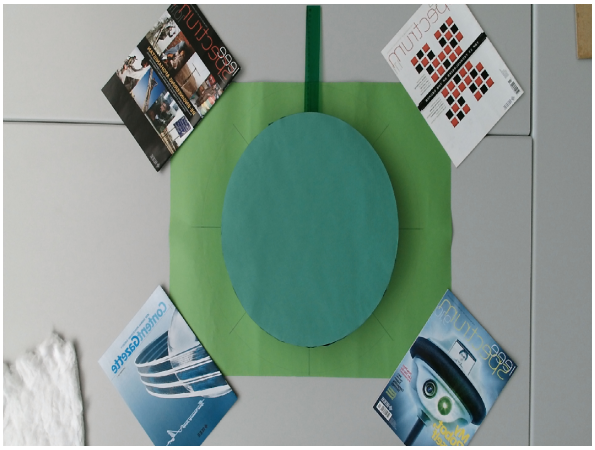


Figura 3.2: Sistema rotatorio de grabación

En la figura 3.2 observamos en detalle el sistema rotatorio. La plataforma se sitúa en una cartulina en la cual están marcados los ángulos entre 0^{o} a 315^{o} en pasos de 45^{o} entre ellos. De esta forma y con la ayuda de la regla situada en la base de la plataforma podemos rotar el material capturándolo así en los 8 ángulos analizados.

3.3. Sistema de captura.

Para la grabación se ha utilizado como *hardware* el sensor Microsoft Kinect v2 descrito en la sección 2.2.2 [1] que tiene la posibilidad de obtener los tres tipos de imágenes deseados. El diseño del sistema de captura se basa en un *software* de grabación que puede describirse mediante el diagrama de flujo de la figura 3.3 .

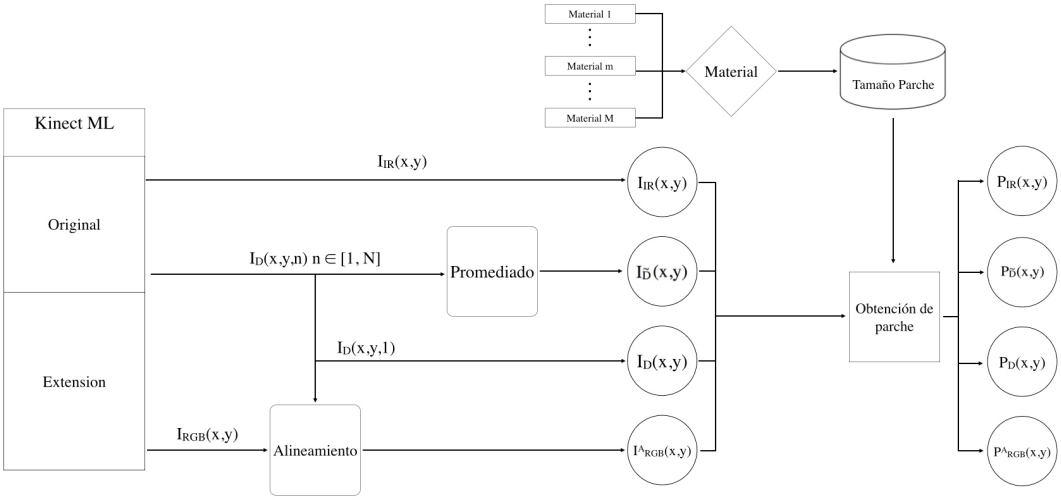


Figura 3.3: Diagrama de flujo del sistema de grabación

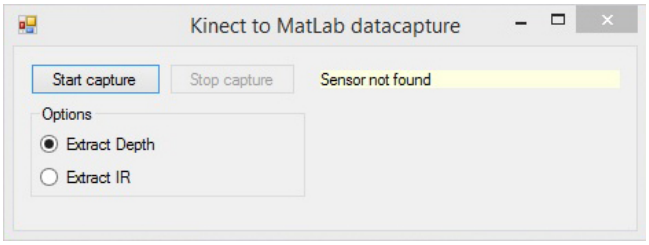


Figura 3.4: Interfaz básica Kinect ML

Como se observa en diagrama de flujo, el *software* de grabación está formado por varios módulos independientes entre sí:

- Kinect ML: Se trata de un software de captura de imágenes de profundidad, infrarrojos y color mediante Kinect. Estas imágenes serán usadas posteriormente en el sistema desarrollado.
- Postprocesado: Esta etapa se realiza mediante Matlab y realiza diversos procesados sobre las imágenes capturadas para la creación de nuevas imágenes para el análisis.
- Generación de parches: El sistema desarrollado trabaja únicamente con parches obtenidos de las imágenes completas a fin de aislar el material. Este módulo realiza todo el proceso de selección y obtención de los diversos parches. Su funcionamiento esta explicado en detalle en el diagrama de flujo de la figura 3.3.

En esta sección se desarrollara cada bloque de este diagrama en detalle para explicar su funcionamiento y su función en el sistema conjunto de grabación.

3.3.1. Programa original

Este programa denominado Kinect ML fue diseñado por [25] y ha constituido la base del software de grabación. El programa original se ha actualizado, añadiendo nuevas funcionalidades exclusivamente para este trabajo. La interfaz inicial del programa era la que podemos observar en la figura 3.4.

En un principio las funcionalidades originales incluidas en la versión inicial del programa eran las siguientes:

1. Obtención de una secuencia de imágenes de profundidad $I_D(x, y, n)$, $n \in [1, N_D]$
2. Obtención de una secuencia de imágenes de infrarrojos $I_{IR}(x, y, n)$, $n \in [1, N_{IR}]$

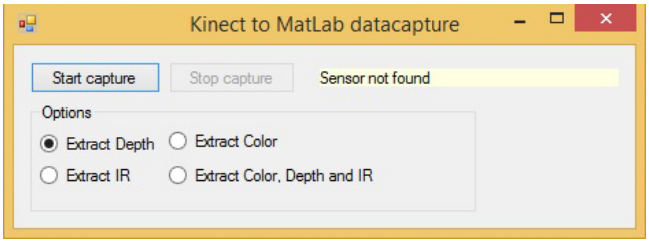


Figura 3.5: Interfaz modificada KinectML

Siendo N_D y N_{IR} el número de imágenes en la secuencia. Ambos parámetros son configurables durante el proceso de captura. En particular vienen determinados por la tasa de servicio efectiva (que depende de la cámara, la máquina utilizada para capturar los datos y de su sistema operativo). En términos prácticos, el programa sirve a la máxima tasa de servicio posible en el periodo comprendido entre la petición de servicio (*Start capture*) y la petición de parada (*Stop capture*) en la figura 3.4.

3.3.2. Actualización. Obtención Imagen en color

Sin embargo este programa no tenía todas las funcionalidades necesarias para poder capturar la base de datos deseada. Por esta razón hubo que modificar el código fuente del programa para añadir nuevas funcionalidades, obteniendo así una nueva versión de la interfaz (Figura 3.5).

La primera mejora y fundamental, la posibilidad de obtener imágenes en color. Dado que la segunda generación de Kinect es relativamente nueva nadie actualmente había hecho público código que sirviera las imágenes en color que suministra la cámara. Por lo tanto hubo que dar un primer paso en este aspecto añadiendo esta modificación, permitiendo servir imágenes en color $I_{RGB}(x, y)$ mediante Kinect ML. Este proceso está detallado ampliamente en el anexo 7.2.

Adicionalmente, se incluye una segunda modificación del programa para realizar una optimización del software. Se ha implementado la funcionalidad para que Kinect ML no sirva imágenes constantemente. En particular, los procesos de servicio de datos se diseñan para una imagen de color $I_{RGB}(x, y)$ y una imagen de infrarrojos $I_{IR}(x, y)$. La tasa de servicio de la información de profundidad será estudiada más adelante en la sección 3.3.3

Esto se ha realizado puesto que la tasa de servicio efectiva del flujo de imágenes de color es mucho menor que la que se obtiene para la profundidad y la información de infrarrojos, puesto que además de componerse de la información de tres canales, la resolución de la imagen capturada por la cámara RGB es mucho mayor que la de profundidad o infrarrojos (ver tabla 2.2 en la sección 2.2). En el ámbito del trabajo fin

de grado que se presenta, esto no va a suponer un problema, puesto que el prototipo diseñado trabaja sobre imágenes estáticas.

3.3.3. Promediado de imágenes de profundidad

Como hemos mencionado anteriormente en el programa se limitó la extracción de una secuencia de imágenes a tan solo una imagen de color $I_{RGB}(x, y)$ y una imagen de infrarrojos $I_{IR}(x, y)$. Sin embargo este proceso no aplica a la información de profundidad, o al menos no totalmente. En [26] se sugiere un método para aumentar la calidad de la profundidad obtenida. La idea es obtener una imagen de profundidad en la cual podamos eliminar el mayor numero de errores procedentes de la captura de los datos y de esta forma mejorar el siempre ruidoso proceso de captura de la profundidad. Este método se denomina *Depth Interpolation* y consiste en capturar N_D imágenes de profundidad: $I_D(x, y, n), n \in [1, N_D]$, con N_D fijado por el usuario, y no dependiente de la tasa de servicio efectiva (como en la sección 3.3.2), para luego hacer el promediado entre ellas para obtener una imagen interpolada temporalmente $I_{\tilde{D}}(x, y)$.

Cuando el proceso de captura de la información de profundidad falla (en zonas oscuras, metálicas o de reflexión especular), el sensor Kinect devuelve valores indeterminados (*NaN: Not-a-number*). Con el objetivo de eliminar su influencia en el promediado se deberá incluir un factor de corrección en su cálculo. En particular, el promediado se calcula:

$$I_{\tilde{D}}(x, y) = \frac{\sum_{n=1}^{N_D} I_{\tilde{D}}(x, y) + I_D(x, y, n)}{V(x, y)} \tag{3.1}$$

siendo $V(x, y)$,

$$V(x, y) = N_D - \sum_{n=1}^{N_D} T(I_{\tilde{D}}(x, y, n)) \tag{3.2}$$

y $T(I_{\tilde{D}}(x, y, n))$ una función que devuelve 1 para posiciones x_0, y_0, n_0 donde la imagen de profundidad tiene valores indeterminados:

$$T(I_{\tilde{D}}(x_0, y_0, n_0)) = \begin{cases} 1, & I_{\tilde{D}}(x_0, y_0, n_0) = NaN \\ 0, & resto \end{cases} \tag{3.3}$$

En el desarrollo de este trabajo fin de grado se ha utilizado $N_D = 20$, por coincidir este valor con la tasa de servicio efectiva aproximada para un segundo de servicio. Adicionalmente, para mantener la información original capturada, y puesto que los

fallos de captura de la profundidad pueden ser indicativos del material grabado, se incluye la primera imagen de profundidad de la serie $I_D(x, y) = I_D(x, y, 1)$ como una imagen adicional a analizar a lo largo del sistema desarrollado.

3.3.4. Calibración del color y la profundidad

Las imágenes de profundidad $I_D(x, y)$ y de infrarrojos $I_{IR}(x, y)$ no tienen la misma resolución que la de color $I_{RGB}(x, y)$ y lo que es más importante, no están espacialmente alineadas con ésta. Esto supone un grave problema puesto que el prototipo propuesto utiliza de manera conjunta información de los diferentes tipos de imágenes, por lo tanto, la información de descripción tiene que representar una zona espacial idéntica de cada uno de las imágenes para poder describir el mismo área espacial del material. Esto es imposible teniendo imágenes de diferente resolución y que han sido captadas por sensores desplazados físicamente o de diferente naturaleza dentro de la Kinect.

Por esta razón se incluyó en el sistema de grabación de la figura 3.3 un nuevo bloque de calibración cuya funcionalidad fuera tomar como entrada una imagen en color $I_{RGB}(x, y)$ y una en profundidad $I_D(x, y)$ y realizar el proceso para obtener a la salida una imagen en color $I^A_{RGB}(x, y)$ de la misma resolución y completamente alineada con respecto a la imagen en profundidad $I_D(x, y)$ (y a la de infrarrojos, que ya está alineada con ésta). Denominamos a este módulo, módulo de Alineamiento.

El proceso se realizará en un bloque constituido en el sistema Kinect ML. Este sistema tomará las imágenes previamente comentadas como entrada. Haciendo uso tanto de las matrices de calibración internas de la cámara, como de las distancias focales de ambos sensores como de la posición física que ocupan los píxeles en ambas cámaras generará un mapeo de un valor de color RGB para cada pixel presente de la imagen de profundidad. De esta forma estaremos asignando a cada punto de una nueva imagen del mismo tamaño de la profundidad un valor de color (por lo que ya tendremos resuelto el problema de las diferentes resoluciones). La descripción del proceso de funcionamiento del módulo de Alineamiento se incluye en el anexo 7.3.

3.3.5. Imágenes obtenidas

Por lo tanto al final del sistema resumido en el diagrama de la figura 3.3 las imágenes que obtenemos finalmente mediante el software de grabación así como sus resoluciones y el numero de frames están recogidos en la tabla 3.2.

Verision Kinect ML	Caracteristica	Version 1	Version 2	Version (Matlab)
Profundidad $I_D(x,y,n)\|_{n=1}$	Numero de Frames	Ilimitados	20	20
	Resolucion	424x512p	424x512p	424x512p
Infrarrojos $I_{IR}(x,y)$	Numero de Frames	Ilimitados	1	1
	Resolucion	424x512p	424x512p	424x512p
Color $d_{RGB}(x,y)$	Numero de Frames	-	1	1
	Resolucion	-	1080x1920p	1080x1920p
Color Alineado $I_{RGB}(x,y)$	Numero de Frames	-	1	1
	Resolucion	-	424x512p	424x512p
Profundidad Interpolada $I_{D\sim}(x,y)$	Numero de Frames	-	-	1
	Resolucion	-	-	424x512p

Tabla 3.2: Tabla resumen con las imágenes obtenidas del software de grabación

Podemos observar la imagen en color alineada $I^A_{RGB}(x,y)$ en la figura 3.6. Esto supone que se pueden utilizar de forma conjunta tanto la imagen en profundidad $I_D(x,y)$, la imagen $I_{IR}(x,y)$ de infrarrojos como la imagen $I^A_{RGB}(x,y)$ de color para la extracción de características mediante descriptores.

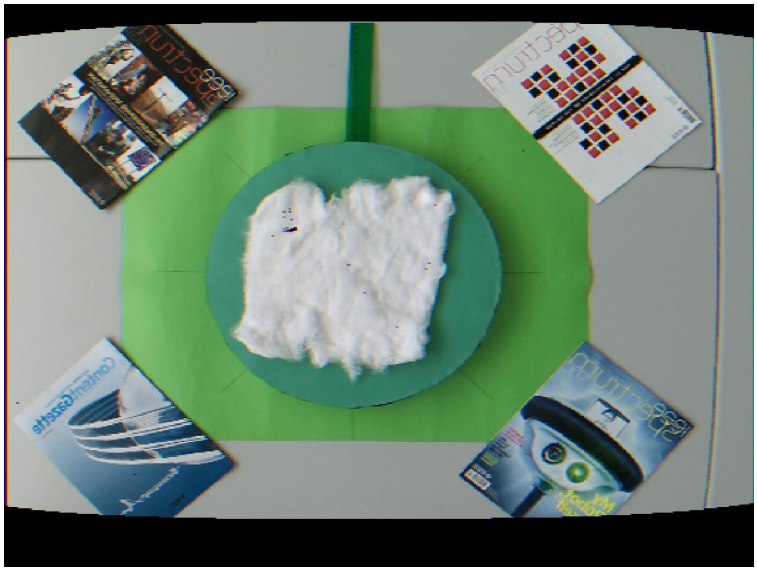
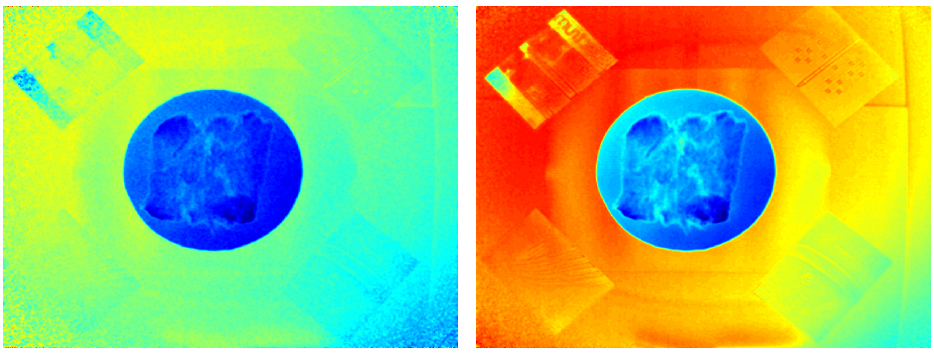
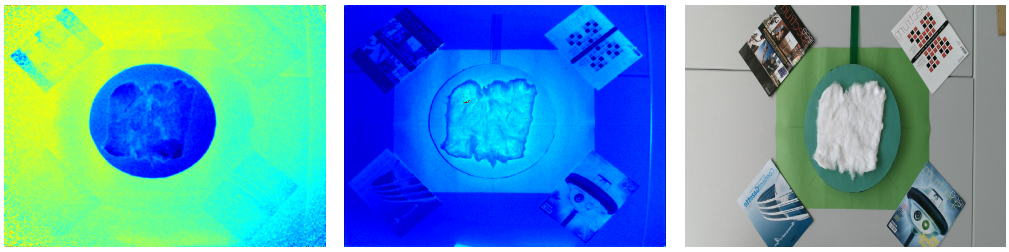


Figura 3.6: Imagen en color alineada $I^A_{RGB}(x,y)$



(a) Frame Depth $I_D(x, y)$ (b) Frame Interpolada $I_{\tilde{D}}(x, y)$

Figura 3.8: Diferencias entre las imágenes de profundidad en el proceso de "Depth Interpolation"



(a) Frame Depth $I_D(x, y)$ (b) Frame IR $I_{IR}(x, y)$ (c) Frame Color $d_{RGB}(x, y)$

Figura 3.7: Conjunto de imágenes obtenidas con Kinect ML versión 2

Como observamos en las imágenes obtenidas mediante el sistema de grabación desarrollado tenemos capturas de materiales sobre nuestro entorno de grabación. Sin embargo, el sistema va a necesitar describir de la forma más precisa los materiales grabados y por ellos será necesario aislarlos de este entorno. Para ello se propone el uso del ultimo modulo del diagrama de flujo de la figura 3.3, una generación de parches de cada imagen capturada.

3.4. Generación semiautomática de parches de material

Como se ha comentado anteriormente la funcionalidad de esta sistema no es utilizar toda una imagen de la base de datos para que el sistema trabaje con ella, si no la de extraer cierta información de un área espacial concreta. Esto nos va a proporcionar dos ventajas.

La primera aislar el material de todo el entorno de grabación. La cámara tiene unas limitaciones de distancia por lo que será imposible grabar una imagen donde

solo este presente el material y por lo tanto tendremos que seleccionar una zona para su correcta descripción.

La segunda será una ventaja implícita a la primera (aunque no buscada) y es que al reducir las dimensiones de nuestra imagen el sistema será computacionalmente más eficiente.

Por estas razones, se propone aislar de cada imagen analizada una zona espacial. Este proceso creará nuevas imágenes, de menor tamaño que las originales, a las que denominaremos parches. En particular, para cada conjunto de imágenes de entrada $\{I_j(x,y),\ j = \{D, \tilde{D}, IR, R, G, B\}\}$ se obtendrá un conjunto de parches $\{P_j(x,y),\ j = \{D, \tilde{D}, IR, R, G, B\}\}$ todos del mismo tamaño y describiendo el mismo área espacial.

Selección del centro del parche

El proceso de selección de parches comienza con la selección del centro del mismo. Este va a ser el primer paso clave de la selección del parche. El centro de coordenadas $(Centro_x, Centro_y)$ se debe mantener fijo a lo largo de la selección de los parches en la base de datos (cada versión tendrá un centro diferente por haber sido grabadas en sesiones diferentes). El proceso de selección del centro va a realizarse mediante Matlab para obtener el centro de la plataforma giratoria de la figura 3.2 siguiendo el diagrama de flujo de la figura 3.9. Este sistema de calibración del centro va a suponer que durante el proceso de grabación los materiales se colocaron correctamente en el centro de la plataforma y que por tanto seleccionando ese punto podremos maximizar el tamaño del parche. Nótese que el proceso puede definirse como semi automático, puesto que el usuario tiene que indicar un punto en la imagen para cada versión de la base de datos, es decir, en total dos puntos.

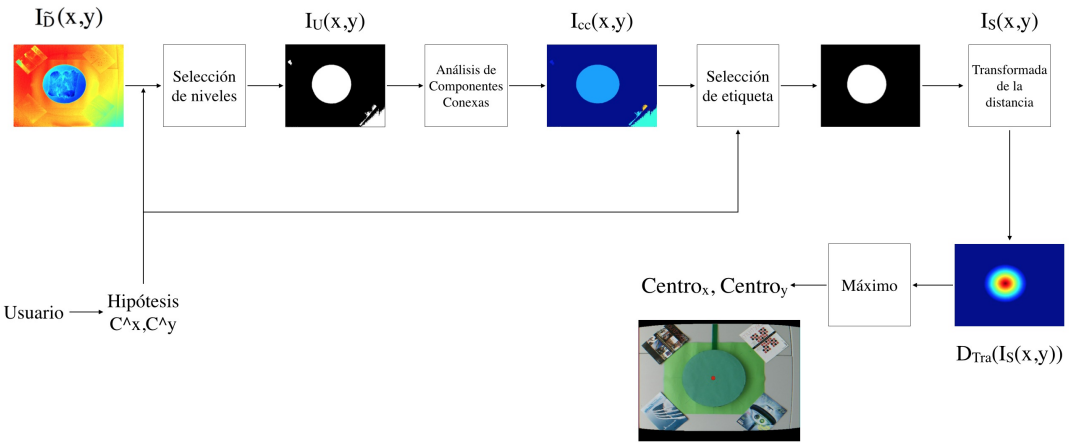


Figura 3.9: Diagrama de flujo del método de generación semiautomática de parches

Los módulos desarrollados en el sistema de selección del centro del parche son:

- Hipótesis del usuario: Se trata de la primera elección del modulo. El usuario formula una hipótesis $(\widehat{C}_x, \widehat{C}_y)$ sobre el lugar donde puede encontrarse el centro de la plataforma deseado; teniendo una aproximación cercana tanto de la posición espacial como del valor en la imagen $I_{\tilde{D}}(\widehat{C}_x, \widehat{C}_y)$ de profundidad interpolada.

- Selección de nivel de la profundidad: Una vez generada la hipótesis por lo tanto podemos utilizar el valor $I_{\tilde{D}}(\widehat{C}_x, \widehat{C}_y)$ para generar una mascara de isoprofundidad $I_U(x, y)$ de la siguiente forma.

$$I_U(x, y) = \left[I_{\tilde{D}}(x, y) - I_{\tilde{D}}(\widehat{C}_x, \widehat{C}_y) \right] \leq Umbral \tag{3.4}$$

  siendo $Umbral$ un valor que nos permita segmentar de la mejor manera posible la totalidad de la plataforma

- Análisis de componentes conexas: Una vez tenemos la imagen binaria $I_U(x, y)$ podemos realizar un análisis de componentes conexas. Con este procedimiento marcaremos una imagen binaria con diferentes etiquetas para las componentes conexas del mismo objeto, de este modo todos aquellos pixeles activados en la mascara y que se encuentren conectados entre ellos tendrán asignado el mismo valor. De esta forma obtendremos la imagen $I_{cc}(x, y)$.

- Selección de la plataforma: Una vez $I_{cc}(x, y)$ y haciendo uso de la hipótesis de usuario$(\widehat{C}_x, \widehat{C}_y)$ podemos obtener la etiqueta asignada por las componentes conexas para el conjunto de pixeles que forman la plataforma. Si denominamos esta etiqueta como $I_{cc}(\widehat{C}_x, \widehat{C}_y)$ podemos obtener la máscara que describe la extensión de la plataforma $I_s(x, y)$ como sigue

$$I_s(x, y) = \begin{cases} I_{cc}(x, y) = I_{cc}(\widehat{C}_x, \widehat{C}_y), & 1 \\ resto, & 0 \end{cases} \tag{3.5}$$

- Transformada de distancia: De esta forma tendremos ya la plataforma correctamente segregada en la imagen binaria $I_s(x, y)$. Aplicando la transformada de la distancia asignaremos a cada pixel activo de $I_s(x, y)$ un número que es la distancia entre ese pixel y el pixel con valor igual a cero más cercano de la imagen obteniendo de esta manera $D_{Tra}(I_s(x, y))$.

- Localización del centro: El ultimo paso será por tanto obtener el centro $(Centro_x, Centro_y)$ como el máximo global de la transformada $D_{Tra}(I_s(x, y))$ obteniendo por tanto sus coordenadas horizontal y vertical.

Durante este proceso de extracción del centro se han realizado tres asumpciones:

1. La cámara está posicionada en un ángulo aproximado de $90^{\underline{o}}$ respecto al plano de la plataforma
2. La plataforma está elevada sobre el plano donde se apoya
3. La posición relativa entre la cámara y la plataforma se mantiene inalterada durante la grabación de toda la base de datos.

Selección secuencial de parches

Una vez tenemos seleccionado un centro común para toda la base de datos hay que seleccionar los parches. Evidentemente no todos los materiales grabados son del mismo tamaño y como el objetivo fundamental es maximizar el tamaño del parche no podremos imponer un tamaño común a todos ellos, en la medida de lo posible pondremos el mismo valor, pero habrá situaciones en las que que el material será tan pequeño que deberemos de reducir el tamaño de su parche. Para la selección del tamaño del parche se ha aplicado un método secuencial en el cual se fija un tamaño inicial de parche cuadrado de 141x141 (centrado en el centro extraído en el apartado anterior) y a partir de ahí se ira disminuyendo el tamaño y probando empíricamente hasta encontrar los mejores posibles.

Este proceso de selección de parches va a estar influenciado por dos factores.

- Material
- Ángulo de rotación en el plano de la plataforma.

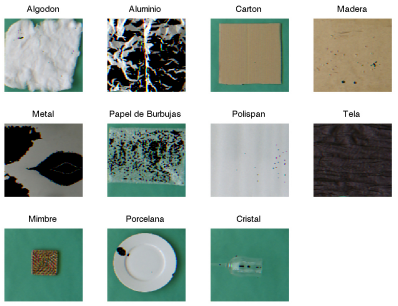
Estos factores van a ser fundamentales puesto que van a determinar el tamaño del parche final. Podemos observar una primera aproximación de tamaño de parche de 141x141 en la figura 3.10a

Como podemos observar, en la figura 3.10a, el tamaño elegido no es el apropiado para todos los materiales, puesto que en muchos casos no hemos aislado completamente el material del entorno. Sin embargo en lo objetos de mayor tamaño relativo como la madera, el metal o la tela, el área de descripción sólo contiene información del material. Existen algunos parches que están cerca de estar completamente cubiertos, como por ejemplo el algodón y el papel de burbujas pero que parecen necesitar un tamaño algo menor. Observando esta necesidad, se propone utilizar un tamaño 81x81, tal y cómo se indica en la figura 3.10b

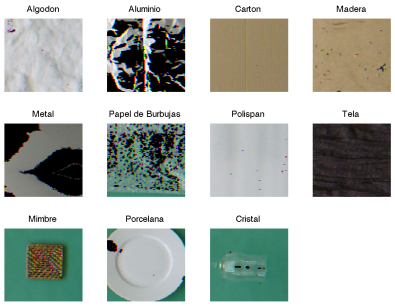
En este caso ya tenemos siete de los once parches correctamente seleccionados pero quedan cuatro materiales en los cuales habrá que reducir indudablemente el tamaño. Siguiendo con el método de evaluación subjetiva, llegamos a la figura 3.10c

Lamentablemente, el ángulo de rotación del material en la plataforma también va a influir en la selección del tamaño. Si mantuviéramos estos tamaños de parche obtendríamos la imágenes correspondientes a la figura 3.10d

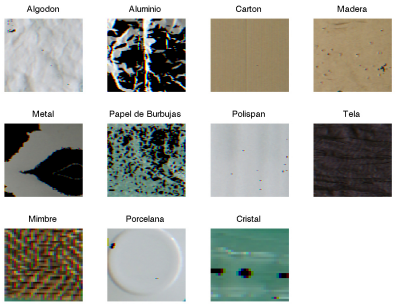
Es decir los tamaños seleccionados funcionan correctamente en los ángulos perpendiculares a los ejes del circulo pero no en los de las diagonales (por se parches cuadrados y no circulares). La solución al problema es reducir de nuevo el tamaño de los parches hasta conseguir de nuevo ajustarlos y maximizarlos. De esta forma obtenemos los parches de la figura 3.10e .



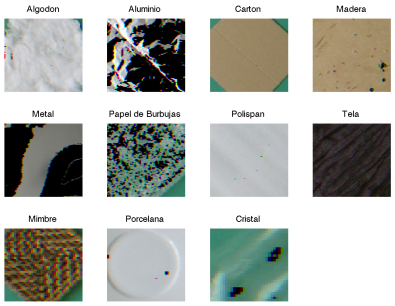
(a) Tamaño de parche 141x141



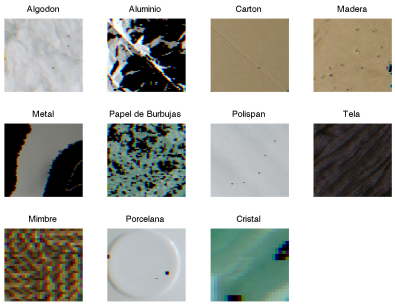
(b) Tamaño de parche 81x81



(c) Tamaños seleccionados sobre angulo 0º



(d) Tamaños seleccionados sobre 45º



(e) Tamaños finales

Figura 3.10: Proceso de selección secuencial de parches

<table>
<tr><td></td><td colspan="2">Tamaño de Parche</td></tr>
<tr><td>Base de datos</td><td>Version 1</td><td>Version 2</td></tr>
<tr><td>Algodon</td><td rowspan="5">81x81 px</td><td>93x93 px</td></tr>
<tr><td>Aluminio</td><td>91x91 px</td></tr>
<tr><td>Carton</td><td rowspan="2">101x101 px</td></tr>
<tr><td>Madera</td></tr>
<tr><td>Metal</td><td>61x61 px</td></tr>
<tr><td>Papel de burbujas</td><td>63x63 px</td><td rowspan="3">101x101 px</td></tr>
<tr><td>Poliespan</td><td rowspan="2">81x81 px
81x81 px</td></tr>
<tr><td>Tela</td></tr>
<tr><td>Mimbre</td><td>31x31 px</td><td>51x51 px</td></tr>
<tr><td>Porcelana</td><td>75x75 px</td><td>61x61 px</td></tr>
<tr><td>Cristal</td><td>23x23 px</td><td>75x75 px</td></tr>
</table>

Tabla 3.3: Tabla resumen de los tamaños de parche usados en la base de datos

Evidentemente dada la selección de dos objetos diferentes por cada material analizado, es decir, dadas las dos versiones de la base de datos, tendremos en algunas ocasiones dos tamaños diferente por objeto. Los tamaños de parche finalmente seleccionados para todas las imágenes de las dos versiones de la base de datos quedan resumidos en la tabla 3.3.

3.5. Composición de la base de datos

La base de datos creada se compone de conjuntos de parches cada uno representando la información de dos objetos para cada uno de los once materiales, cada uno de ellos capturado con dos iluminaciones diferentes y bajo ocho ángulos de rotación sobre el plano de captura.

Si definimos una instancia de material como el conjunto de parches que lo describen en cada imagen analizada: $\{P_j(x,y),\ j = \{D, \tilde{D}, IR, R, G, B\}\}$, el proceso de captura resultan en un total de 352 instancias de material que constituyen la base de datos creada. La base de datos así constituida se hará pública para la comunidad investigadora mediante la página web del grupo de investigación al que se adhiere este trabajo fin de grado.

Capítulo 4

Sistema Propuesto

De nuevo y siguiendo con la metodología del capitulo 3 se ha implementado el sistema propuesto mediante el diseño de un diagrama que muestra el flujo por el cual se procede a la identificación automática de materiales y que podemos visualizar en la figura 4.1.

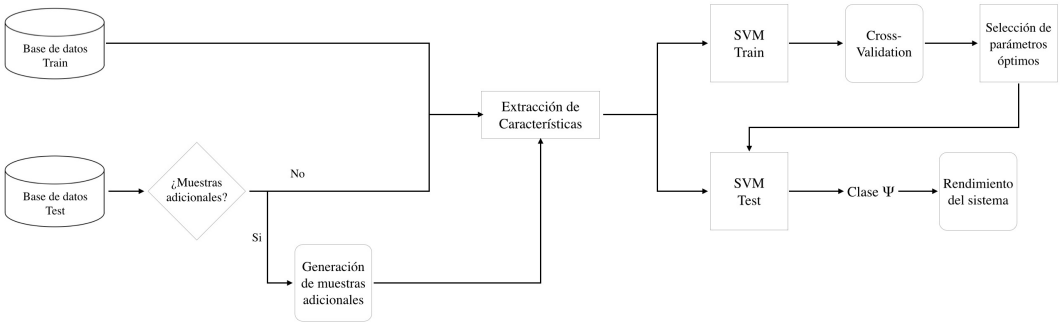


Figura 4.1: Diagrama de flujo del sistema propuesto.

Este sistema va a constar de dos caminos principales: Una rama de entrenamiento (la parte superior del esquema) y una rama de test (su parte inferior). Ambos caminos estarán diseñados de la misma forma pero, exceptuando el bloque de extracción de características, serán diferentes.

- Rama de entrenamiento

  En esta rama del diagrama de flujo entrenaremos un modelo de conocimiento que servirá para predecir muestras futuras. Este proceso parte de una base de datos de instancias de entrenamiento a las que extraeremos, mediante el uso de descriptores, un vector de características, que nos servirá para entrenar una Maquina de Vector Soporte (SVM).

- Rama de test

  Parte de una base de datos de instancias de test. Estas instancias servirán para medir el rendimiento final del sistema. El bloque de extracción de características será idéntico al de la rama de entrenamiento (salvo por un módulo de replicación de datos que describiremos más adelante). Posteriormente se hará uso del modelo de conocimiento generado en la rama de entrenamiento y que permitirá predecir instancias de test mediante SVM. Estas maquinas devolverán una clase $\psi \in \{Algodón, ..., Cristal\}$ para cada vector de características de entrada (es decir, para cada instancia de test) que podremos usar para medir el rendimiento de nuestro sistema.

4.1. Entrenamiento

Como se ha comentado en la introducción esta rama partirá de unas muestras de entrenamiento. Estas muestras se obtendrán de la base de datos descrita en el capitulo 3 mediante su partición aleatoria en dos conjuntos en proporciones variables. El conjunto no utilizado en el entrenamiento constituirá el conjunto de test. La proporción de las muestras totales dedicada a entrenamiento podrá variar y no estará fijado a un valor concreto si no que se evaluará el rendimiento del sistema para diferentes particiones (ver sección 5.2.4).

Una vez tenemos la base de datos de entrenamiento el primer bloque tendrá como finalidad la extracción de características y la creación de un vector de características que permita describir una muestra de un material.

4.1.1. Extracción de características entrenamiento

Selección del descriptor

En la sección 2.3 se describían una serie de formas y aproximaciones de la BDRF para describir una superficie de un material. De todo este conjunto de descriptores se ha realizado una selección y serán los que se utilicen de aquí en adelante a lo largo del trabajo. Se ha seleccionado la descripción en base a HoG por su corroborado potencial en tareas que, aún ajenas al reconocimiento de materiales, comparten el espíritu de descripción de esta tarea. Además la eficiencia en su cálculo y su simplicidad, han sido factores también valorados para su elección.

Se trata de uno de lo descriptores más usados en el reconocimiento de formas en imágenes y por tanto consideramos que puede describir la superficie de material si interpretamos los HoG como una aproximación en dos dimensiones de la BDRF para

estimar la reflectancia de la luz en una superficie mediante gradientes. Sin embargo como se menciono en la sección 2.3.2 una de sus principales desventajas es que la descripción tiende a no ser invariante a la rotación sobre el plano de captura del objeto cuyo material queremos reconocer.Por esta razón se ha decidido implementar en el sistema los dos HoG definidos en [3], GDF-HOG y EigHess-HOG explicados en la sección 2.3.2, que según su descripción son invariantes a cualquier rotación del plano de captura. Como veremos en la sección 5.2.4, esta invarianza a rotaciones es limitada, no completa.

Descriptor por instancia de material

Cada uno de los dos descriptores propuesto tiene dos parámetros de configuración 2.3.2. El primero, N_s, representa el número de celdas horizontales y verticales en las que se dividirá el parche para su descripción por HoG. En este trabajo fin de grado utilizaremos tres valores: $N_s = \{1\,2\,3\}$, por lo que tendremos las divisiones en celdas que representan los casos de la figura 4.2. El segundo los parámetros $Nbins$ define el número de bins sobre el que se construirá el histograma de gradientes orientados en cada celda. En nuestro caso, fijaremos este valor a $Nbins = 9$.

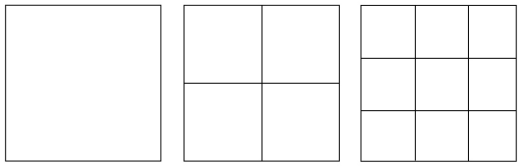


Figura 4.2: Diferentes configuraciones de celdas para $N_s = 3$

Sea $\Upsilon = \{P_j(x,y),\ j = \{D, \tilde{D}, IR, R, G, B\}\}$ una instancia de material a describir se propone extraer y concatenar sus descripciones GDF-HOG y EigHess-HOG para cada parche $P_j(x,y)$ y las tres configuraciones descritas por N_s. Es decir, por ejemplo para un parche $P_j(x,y)$ y el descriptor GDF-HOG se extraerá el histograma del parche completo ($N_s = 1$) y se concatenará con los cuatro histogramas correspondientes a la configuración ($N_s = 2$) y con los nueve histogramas de la configuración ($N_s = 3$, ver figura 4.2 y 4.3). Por otro lado, se repetirá el mismo proceso para el descriptor EigHess-HOG. Generando así en total un vector de dimensión $2(Nbins \sum_{N_s=1}^{3} N_s^2) = 252$ para cada parche $P_j(x,y)$ analizado. Concatenando las descripciones obtenidas para todos los parches que definen la instancia generaremos, finalmente, un vector de descripción de la instancia completa $\mathbf{f}(\Upsilon)$ de dimensión $6 \left[2(Nbins \sum_{N_s=1}^{3} N_s^2) \right] = 1512$.

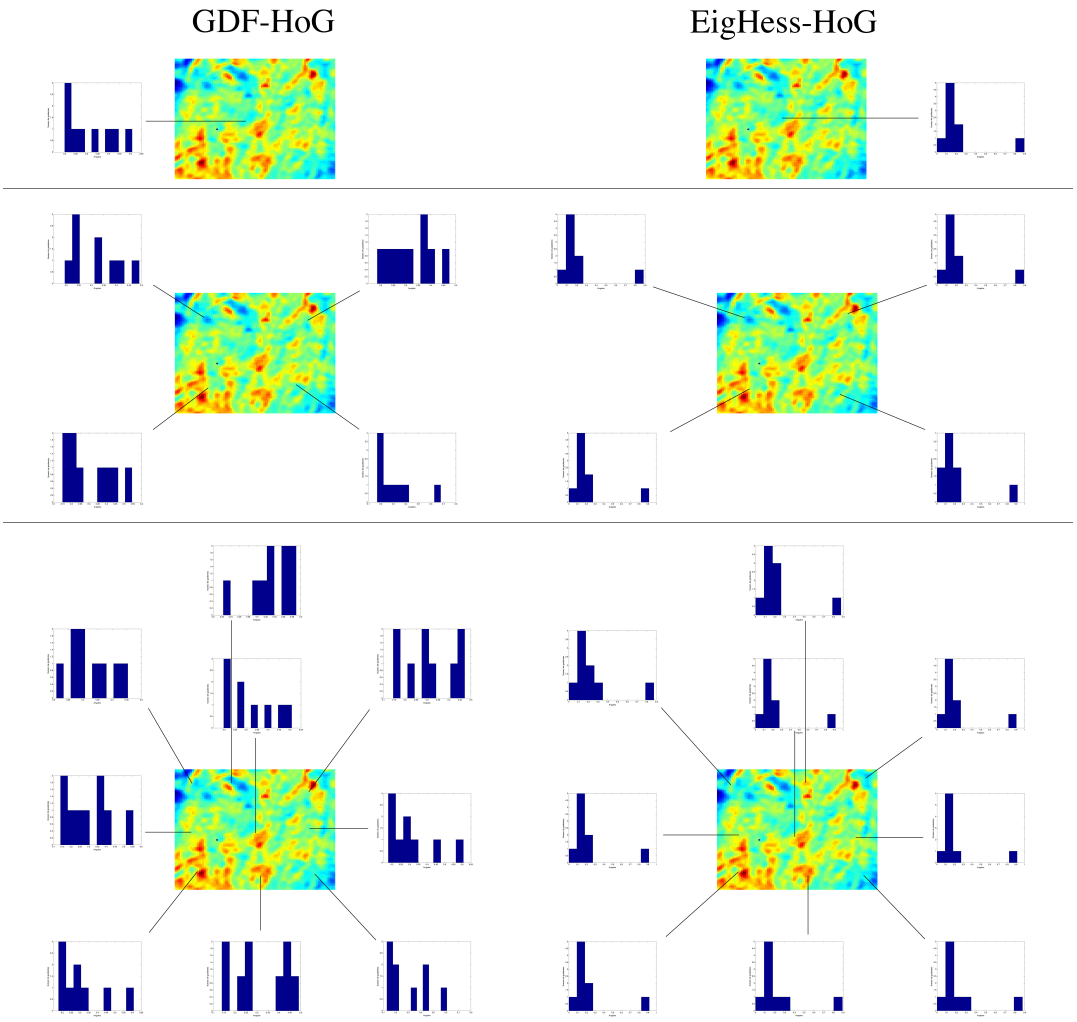


Figura 4.3: Histogramas para GDF-HoG y EigHess-HoG para $N_s = 3$

4.1.2. Generación del modelo de conocimiento mediante SVM Train

Para la utilización de las SVM se ha utilizado la librería LIBSVM para Matlab disponible gratuitamente en Internet [27]. Mediante esta librería podremos generar un modelo de conocimiento a partir de las muestras de entrenamiento y sus descripciones asociadas calculadas mediante la forma explicada en la sección 4.1.1.

Esta será la etapa más importante del algoritmo de clasificación puesto que será aquí donde se genere el modelo que en un futuro nos permitirá predecir muestras futuras a partir de las instancias de entrenamiento. Debemos recordar de lo descrito en la sección 2.4.2 que se tratará de un modelo que mapeará los datos de entrada a una dimensión mayor e incluso infinita y que el algoritmo intentará separarlos en

el espacio mediante hiperplanos de la misma dimensión del modelo y con el mayor margen posible.

Para la realización del modelo vamos a contar con una serie de configuraciones para dos factores que nos permitirán modificar la forma de cálculo del modelo y que estarán desarrolladas en el anexo 7.4.

1. Tipo de SVM

   Podremos elegir entre máquinas de vector soporte de dos tipos diferentes: SVM para problemas de clasificación y SVM para problemas de regresión. Evidentemente al tratarse de clasificación de materiales se usará una SVM de clasificación y en concreto C-SVC (*C-Support Vector Classification*) que tendrá como parámetro de configuración el coste C.

2. Tipo de kernel

   Usaremos un kernel de tipo función de base radial (*Radial Basis Function*, RBF) el cual variará su configuración en función de un parámetro positivo G.

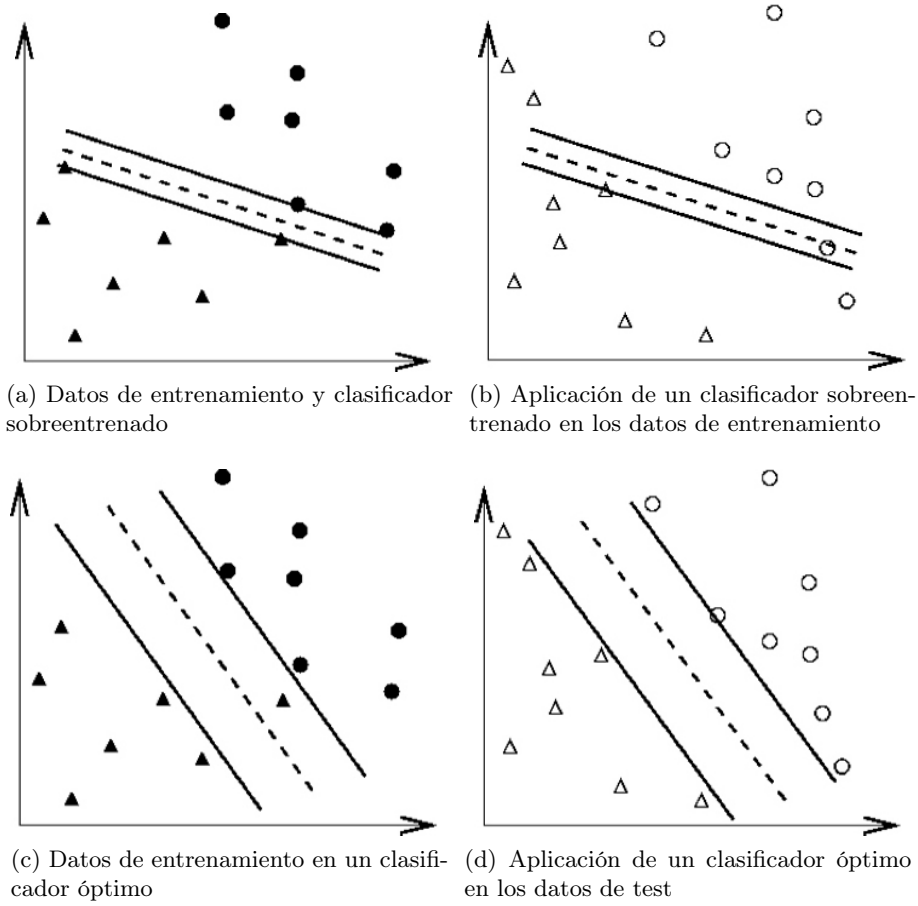
$$K(x_i, x_j) = e^{(-G\|x_i - x_j\|^2)},\ G > 0 \tag{4.1}$$

La elección de este kernel viene dada dado que es el recomendado por la librería apoyándose en el estudio comparativo de [28].

4.1.3. Validación cruzada

Existen dos parámetros configurables (C, G) que definirán junto con los datos de entrada el modelo de conocimiento. Estos parámetros van a depender de la situación en la que se aplique el modelo y por lo tanto no conoceremos de antemano cuáles van a ser los mejores para el sistema propuesto. El objetivo es poder identificar una buena pareja de (C, G) para que podamos predecir con precisión instancias de test. El problema radica en evitar conseguir altos resultados de precisión en la fase de entrenamiento (donde se fijan estos parámetros) puesto que esto podría significar que el modelo estuviera demasiado adaptado a éstos y, por lo tanto, sobreentrenado.

La parte superior de la figura 4.4 representa gráficamente este problema. Como podemos observar en las figuras 4.4a y 4.4b el clasificador no funcionará correctamente puesto que esta sobreentrenado, es decir, se ajusta demasiado a los datos de entrenamiento. Si aplicamos este modelo a los datos de test o bien estos datos son muy parecidos a los de entrenamiento o bien tendremos errores puesto que no podrán separase correctamente ambos conjuntos dado que hemos creado una frontera muy cerrada.



(a) Datos de entrenamiento y clasificador sobreentrenado

(b) Aplicación de un clasificador sobreentrenado en los datos de entrenamiento

(c) Datos de entrenamiento en un clasificador óptimo

(d) Aplicación de un clasificador óptimo en los datos de test

Figura 4.4: Clasificador sobreentrenado vs clasificador óptimo (● y ▲ datos de entrenamiento; ○ y △ datos de test)

Para minimizar el sobreentrenamiento la librería nos permite entrenar el modelo mediante el esquema *v-fold Cross-Validation* (validación cruzada de v hojas). En este modo se dividen los datos de entrenamiento en v subconjuntos del mismo tamaño. De forma secuencial se prueba un subconjunto usando el modelo clasificador generado a partir de los $v - 1$ subconjuntos restantes. En nuestro sistema el número de subconjuntos en los que dividiremos los datos de entrenamiento será $v = 10$. Por lo tanto cada subconjunto será clasificado una vez, y la precisión de la validación cruzada será el porcentaje de datos que se clasifican correctamente.

Las ventajas de la validación cruzada pueden observarse en la parte inferior de la figura 4.4. En las figuras 4.4c y 4.4d no tenemos una clasificación perfecta en entrenamiento pero cuando aplicamos este modelo a los datos de test podemos comprobar que obtenemos mejores resultados al tener una frontera de decisión menos estricta que la obtenida para el entrenamiento sin validación cruzada.

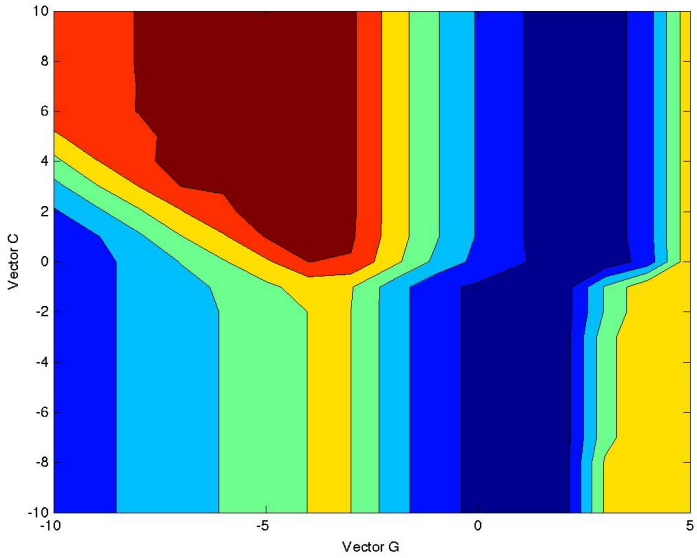


Figura 4.5: Fronteras de decisión mediante Cross-Validation

Por lo tanto si utilizamos este método conjuntamente con un sistema iterativo de variación de parámetros vamos a poder obtener porcentajes de acierto de cada par de parámetros. Estos pares de parámetros definen fronteras de precisión que permiten medir visualmente la precisión obtenida para cada prueba realizada. Un ejemplo de estas fronteras puede observarse en la figura 4.5.

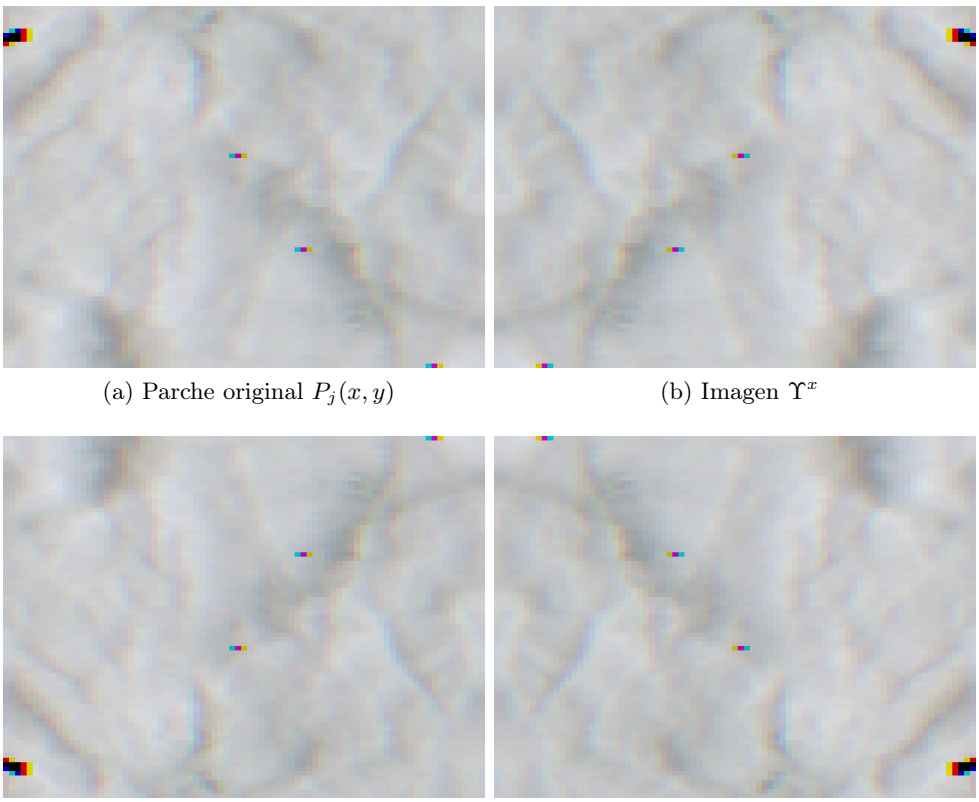
A la vista de las potenciales ventajas de la validación cruzada, utilizaremos esta técnica para estimar los parámetros (C, G) que definen el modelo de conocimiento obtenido en el entrenamiento.

4.2. Test

La segunda rama del sistema propuesto partirá de una base de instancias de test generada mediante los archivos que no hayan sido seleccionados para formar el conjunto de entrenamiento, por lo tanto este conjunto vendrá dado como complementario, por la selección realizara en la sección 4.1.

4.2.1. Extracción de características test

Este bloque será idéntico al explicado en la sección 4.1 para el entrenamiento. Sin embargo en el diagrama de flujo se indica que la entrada a este modulo puede tener una variación con respecto a la rama de entrenamiento. En test se incluye la opción de incluir instancias adicionales obtenidas como versiones de una existente Υ en la



(a) Parche original $P_j(x, y)$ (b) Imagen Υ^x

(c) Imagen Υ^y (d) Imagen Υ^{xy}

Figura 4.6: Ejemplos de imagen original y sus reflexiones

base de datos de test.

En el sistema desarrollado estas nuevas instancias se componen de los mismos parches pero con tres tipos de reflexiones (aplicadas a cada tipo de parche dentro de la instancia):

- Reflexión con respecto al eje x: Υ^x
- Reflexión con respecto al eje y: Υ^y
- Combinación de las dos reflexiones anteriores: Υ^{xy}

Podemos observar ejemplos de estas reflexiones en la figura 4.6.

El objetivo de estos procesos de reflexión es el de comprobar realmente si los algoritmos de descripción utilizados proporcionan invarianza a rotaciones. Para ello se medirá la diferencia entre los porcentajes de acierto obtenidos con y sin este módulo auxiliar.

4.2.2. Test del modelo de conocimiento

Mediante el modulo de test de SVM se predice, utilizando el modelo de conocimiento, para cada vector de características de entrada $\mathbf{f}(\Upsilon)$ una clasificación representada con una etiqueta de clase $\psi(\Upsilon)$ dentro de las once categorías de materiales que tenemos definidas. Adicionalmente la SVM retornará una probabilidad de asignación asociada a esa clasificación $\vartheta(\Upsilon)$. Si los instancias de entrada están etiquetadas, $\psi_{GT}(\Upsilon)$, (como es el caso) podemos calcular el rendimiento del prototipo diseñado obteniendo un porcentaje tanto de acierto como de confusión.

Capítulo 5

Resultados experimentales

5.1. Introducción

Durante esta sección se comentaran en detalle tanto las pruebas realizadas sobre la base de datos del capítulo 3 (tomándola como el conjunto completo de las dos versiones) y el sistema desarrollado en el capítulo 4 como los resultados obtenidos.

El primer aspecto fundamental es la selección de los parámetros y configuraciones usados a lo largo de las pruebas. Estos parámetros se van a a usar a lo largo del sistema y quedarán fijados desde el principio. Como ya se anticipó en el capítulo 4 se utilizarán los siguientes parámetros:

- SVM de tipo clasificatorio C-SVC con kernel de RBF.
- Validación cruzada de $v = 10$ subconjuntos.
- Parámetro $N_s = \{1, 2, 3\}$ para el calculo de los dos tipos de descriptores HoG.

Se han realizado dos tipos diferentes de experimentos que se detallan en las secciones 5.2 y 5.3. Los primeros resultados experimentales se han obtenido mediante la utilización de las imágenes del dataset generado, por lo tanto abarcará todas las pruebas del rendimiento del sistema frente a datos de situaciones controladas de grabación. El segundo tipo de experimentos evaluara el sistema mediante unas imágenes reales que muestran situaciones complejas de reconocimiento mediante un conjunto de materiales capturados en una mesa.

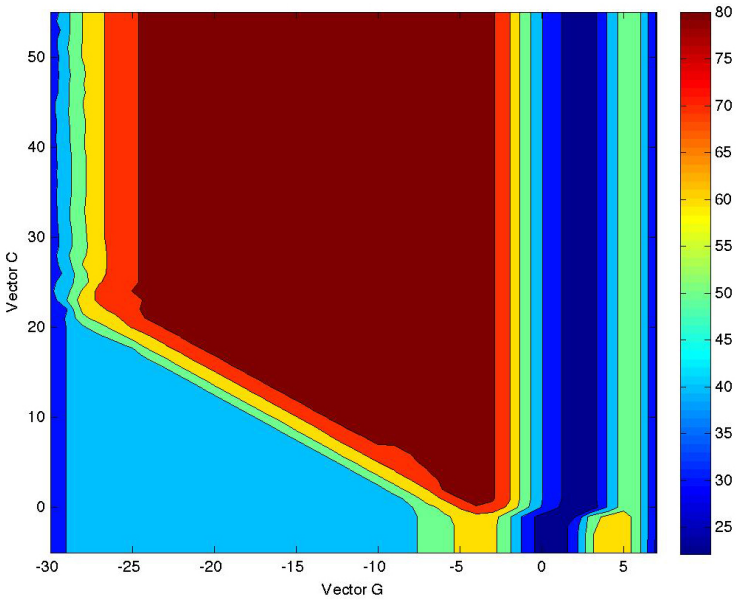


Figura 5.1: Fronteras de porcentaje de acierto usando todo el dataset

5.2. Resultados cuantitativos en escenas controladas

5.2.1. Generación del modelo de conocimiento

Se ha realizado una primera aproximación para el calculo del modelo óptimo mediante el uso de la base de datos al completo y dividiendo ésta aleatoriamente en un 50 % de ficheros de entrenamiento y otro 50 % en ficheros de test. Utilizando validación cruzada se han iterado tanto el parámetro $C \, \epsilon \, [-5 \; 55]$ como el $G \, \epsilon \, [-30 \; 7]$ para obtener de esta forma el par que maximice la precisión del sistema. Las fronteras de acierto obtenidas por cada par (C, G) se incluyen en la figura 5.1

Como podemos observar en la figura 5.1 existe una gran variación en el porcentaje de acierto que conseguimos dependiendo del par de parámetros que elijamos. El valor máximo que logramos es de $85,7955\,\%$ siendo posible de obtener con varios pares de parámetros (C, G). Seleccionamos como parámetros óptimos $C = 10$ y $G = -5$.

5.2.2. Evaluación del modelo de conocimiento

Una vez el modelo ha sido generado, proponemos tres metodologías diferentes de evaluación.

- Metodología 1: En esta metodología solo se tienen en cuenta las instancias originales del dataset, es decir, no se obtienen nuevas muestras adiciones mediante reflexiones (proceso descrito en la sección 4.2.1). Por lo tanto el rendimiento

del sistema está únicamente basado en las imágenes capturadas en el dataset. El porcentaje de acierto obtenido mediante esta metodología se denominará Accuracy 1.

- Metodología 2: De cada fichero de test se han generado otras tres instancias adicionales mediante el proceso de reflexión de la sección 4.2.1. Cada una de los cuatro instancias (Υ, Υ^x, Υ^y, Υ^{xy}) generadas está asignada a la misma clase de material. En esta precisión se calcula si alguna de estas instancias no se ha clasificado correctamente y por tanto cada una de ellas contabilizará como error o acierto. Es decir, se evaluarán independientemente las clases ($\psi(\Upsilon)$, $\psi(\Upsilon^x)$, $\psi(\Upsilon^y)$,$\psi(\Upsilon^{xy})$) devueltas para cada una de las instancias generadas mediante los procesos de reflexión. El porcentaje de acierto obtenido mediante esta metodología se denominará Accuracy 2.

- Metodología 3: Se generan las mismas instancias adicionales mediante reflexión que en la metodología 2. En esta metodología se contabilizarán los errores o aciertos en función de la probabilidad de asignación más alta obtenida para las cuatro instancias. Es decir, se seleccionará la clase ($\psi(\Upsilon)$, $\psi(\Upsilon^x)$, $\psi(\Upsilon^y)$,$\psi(\Upsilon^{xy})$) devuelta para la instancia que lleve asociada la probabilidad de asignación más alta ($\vartheta(\Upsilon)$, $\vartheta(\Upsilon^x)$, $\vartheta(\Upsilon^y)$,$\vartheta(\Upsilon^{xy})$). El porcentaje de acierto obtenido mediante esta metodología se denominará Accuracy 3.

Los resultados obtenidos para el sistema entrenado con división aleatoria de la base de datos completa en un 50 % de ficheros de entrenamiento y otro 50 % en ficheros de test están resumidos en la tabla 5.1

Generación del modelo mediante todo el Dataset							
Modelo de conocimiento					Ficheros Test		
Porcentaje Ficheros	Vector C	Vector G	Parámetros Optimos	Porcentaje Acierto CV	Accuracy 1	Accuracy 2	Accuracy 3
50	-5:55	-30:7	C = 10 G = -5	85,80%	85,23%	66,48%	84,66%

Tabla 5.1: Resultados con 50 % ficheros de entrenamiento usando el dataset completo

Como podemos observar utilizando el 50 % de ficheros del dataset para entrenar conseguimos un porcentaje de acierto en validación cruzada de un 85,80 %. Por otro lado la precisión en la clasificación varía desde un 85,23 % con la metodología 1, un 66,48 % con la metodología 2 y finalmente un 84,66 % con la metodología 3. Existen diferencias notables en los diferentes porcentajes de acierto definidos para este modelo

de conocimiento, las razones por las que esto sucede están explicadas en la sección 5.4.

A fin de evaluar los problemas de clasificación que presenta cada metodología, es interesante incluir información sobre las clasificaciones erróneas que está realizando la SVM. Es decir, qué material está prediciendo erróneamente en los casos de error. Esta información permite ser analizada por medio de las matrices de confusión (50 % ficheros de entrenamiento. $C = 10$ y $G = -5$) incluidas en las figuras 5.2, 5.3 y 5.4.

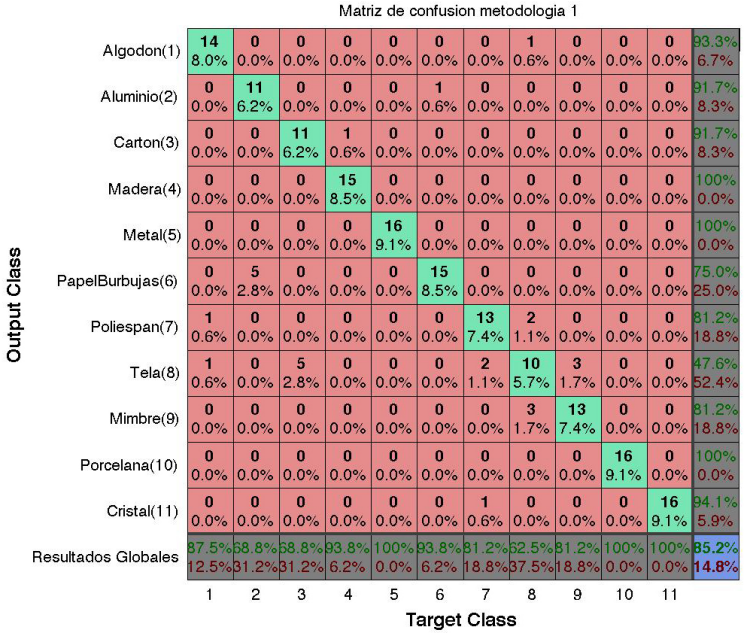


Figura 5.2: Matriz de confusión con metodología 1

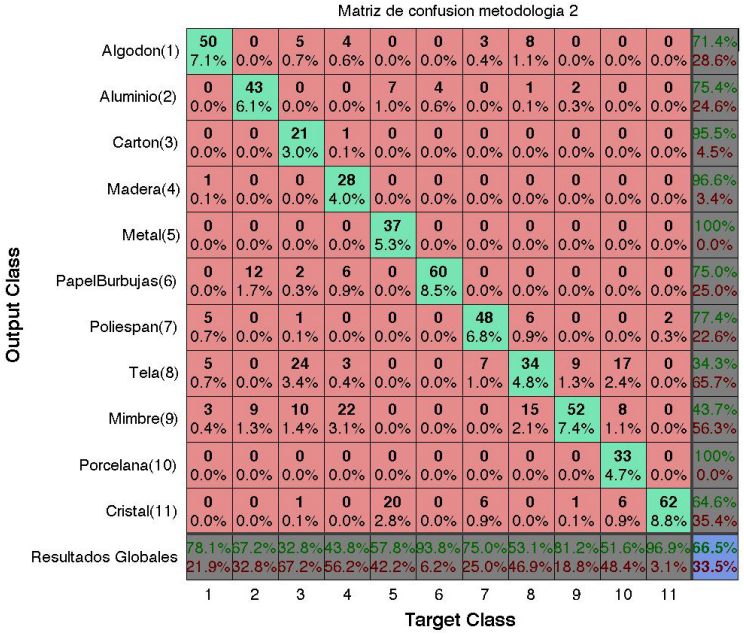


Figura 5.3: Matriz de confusión con metodología 2

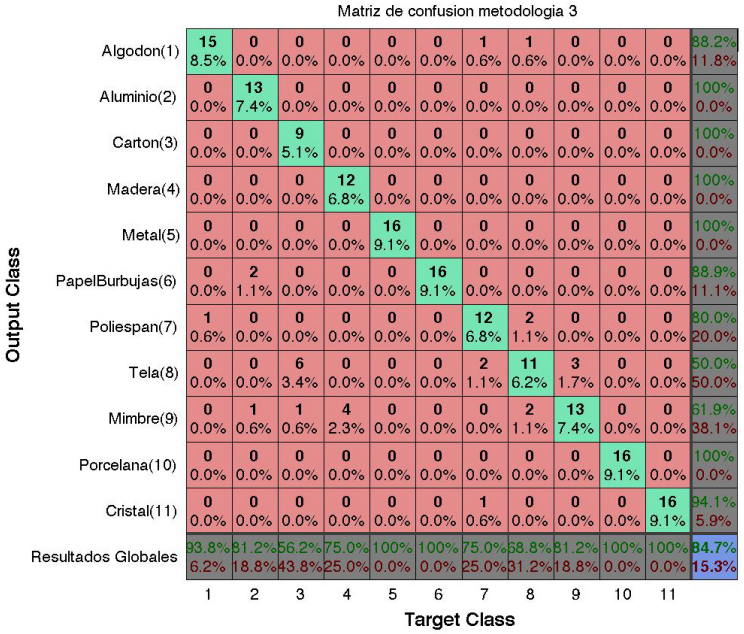


Figura 5.4: Matriz de confusión con metodología 3

5.2.3. Influencia del tamaño del entrenamiento en el rendimiento del sistema

El objetivo de esta sección es el de evaluar la influencia del tamaño del conjunto de entrenamiento en el rendimiento del sistema. Para ello se generarán varios modelos manteniendo el par óptimo seleccionado por validación cruzada ($C = 10$ y $G = -5$) en la sección 5.2.1, pero variando el porcentaje de ficheros que se usan para entrenar. De este modo podemos comprobar si se está utilizando el suficiente número de ficheros para modelar los datos o si por el contrario aumentando o disminuyendo el porcentaje podrían conseguirse porcentajes de acierto superiores.

Para ello, se variará el porcentaje desde el 5 % hasta el 80 % de ficheros de test, obteniendo así los resultados de la tabla 5.2 y la gráfica en la figura 5.5

Modelo de conocimiento C = 10 G = -5		Ficheros Test		
Porcentaje Ficheros (%)	Porcentaje de Acierto CV por SVM (%)	Accuracy 1 (%)	Accuracy 2 (%)	Accuracy 3 (%)
5	31,82	13,64	8,26	10,00
10	33,33	16,30	11,60	15,99
20	66,67	65,73	48,95	66,43
30	77,27	78,51	60,74	80,17
40	76,92	84,21	61,96	82,78
50	85,80	85,23	66,48	84,66
60	88,52	84,62	64,69	86,01
70	86,78	93,64	71,36	88,18
80	89,86	92,42	70,83	87,88

Tabla 5.2: Resultados con variación de porcentaje de ficheros de entrenamiento con $C = 10$ y $G = -5$

5.2.4. Influencia de las versiones de la base de datos: reconocimiento de objetos o de materiales

En la sección 5.2.1 hemos realizado las pruebas dividiendo en subconjuntos de entrenamiento y test de la base de datos completa. El objetivo de esta sección es el de comprobar si el sistema diseñado sería capaz de generar modelos de material y no de objetos utilizando sólo un tipo de objeto para identificar el material. Para ello, se propone realizar la generación de dos modelos de conocimiento los cuales sólo contengan información acerca de una de las dos versiones de la base de datos.

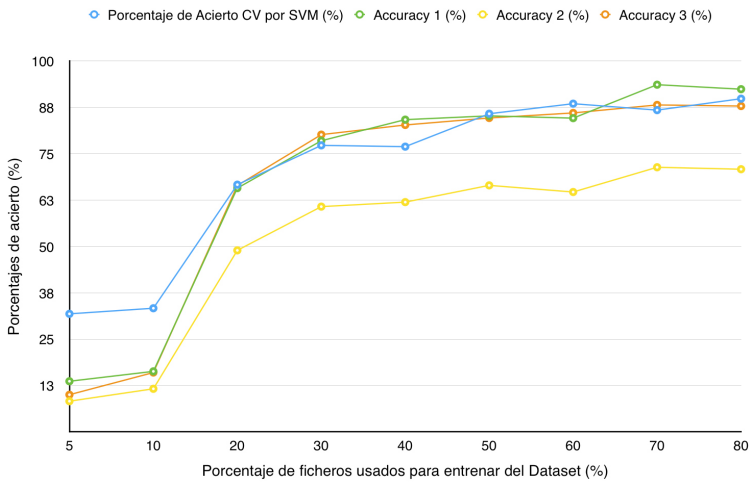


Figura 5.5: Gráfica porcentaje de ficheros entrenamiento frente a precisiones

A continuación se evaluará el rendimiento probando instancias correspondientes a la otra versión.

Los resultados obtenidos entrenando con la versión 1 ($C = 10$ y $G = -5$) y evaluando con la versión 2 se incluyen en la figura 5.6 (fronteras de precisión en la validación cruzada), la tabla 5.3 (precisión en la detección con cada metodología propuesta) y las figuras 5.7, 5.8 y 5.9 (matrices de confusión).

Por otro lado, los resultados obtenidos entrenando con la versión 2 ($C = 10$ y $G = -5$) y evaluando con la versión 1 se incluyen en la figura 5.10 (fronteras de precisión en la validación cruzada), la tabla 5.4 (precisión en la detección con cada metodología propuesta) y las figuras 5.11, 5.12 y 5.13 (matrices de confusión).

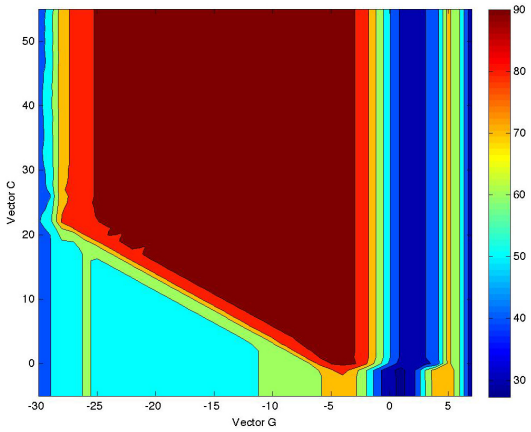


Figura 5.6: Curvas de precisión por CV generando el modelo con el primer subconjunto y evaluando con el segundo

Generación del modelo mediante el primer conjunto de objetos

Modelo de conocimiento					Ficheros Test		
Porcentaje Ficheros	Vector C	Vector G	Parámetros Optimos	Porcentaje Acierto CV	Accuracy 1	Accuracy 2	Accuracy 3
50	-5:55	-30:7	C = 4 G = -5	93,75%	25,00%	24,72%	25,00%

Tabla 5.3: Resultados de precisión. Modelo generado con el primer subconjunto de objetos y evaluado con el segundo

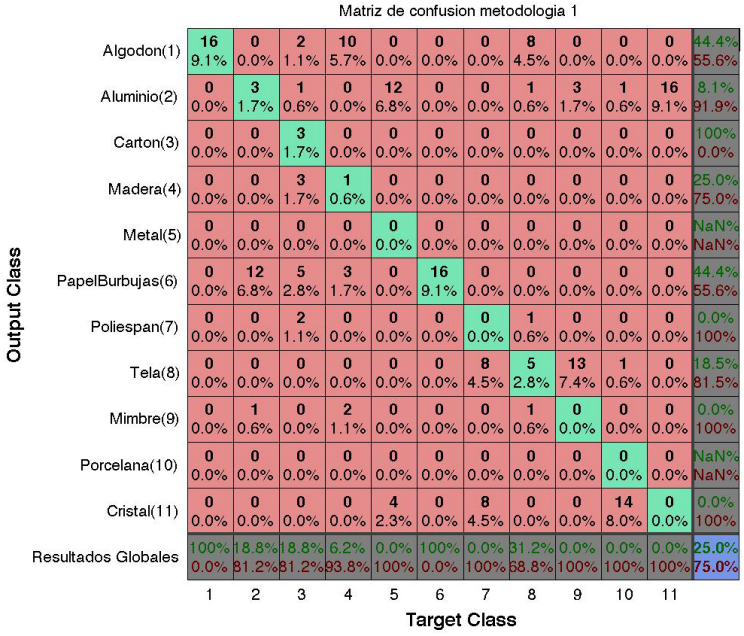


Figura 5.7: Matriz de confusión entrenando con la versión 1 (metodología 1)

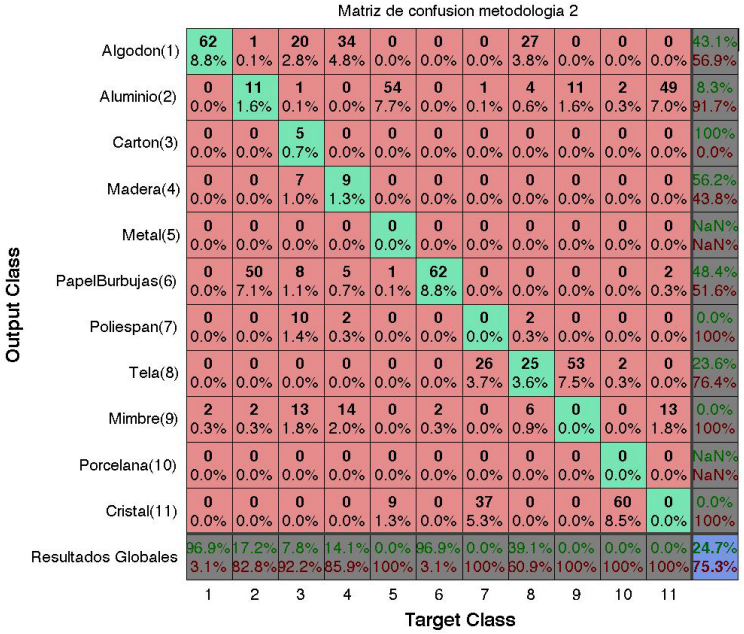


Figura 5.8: Matriz de confusión entrenando con la versión 1 (metodología 2)

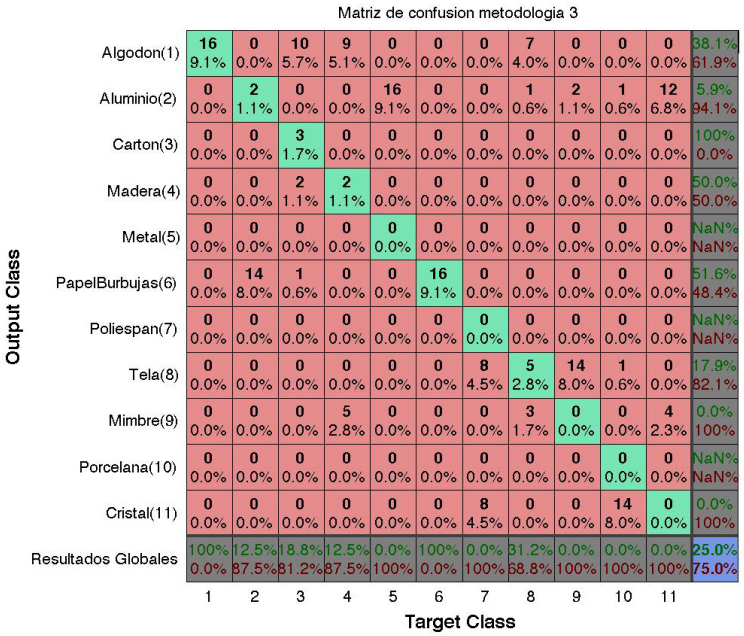


Figura 5.9: Matriz de confusión entrenando con la versión 1 (metodología 3)

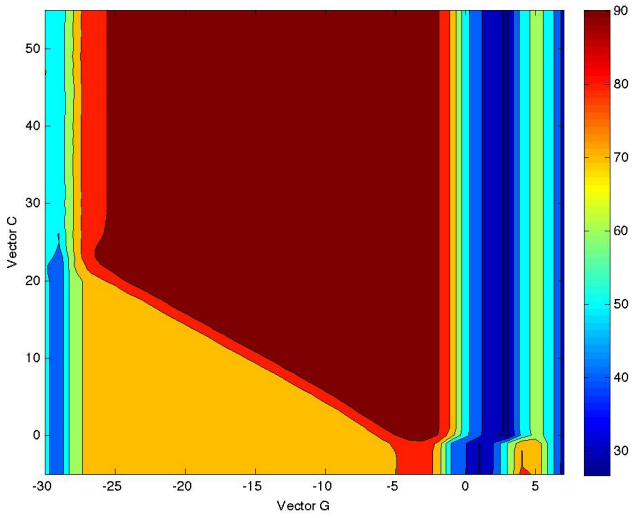


Figura 5.10: Curvas de precisión por CV generando el modelo con el segundo subconjunto y evaluando con el primero

Resultados generando el modelo con el segundo conjunto de objetos

Modelo de conocimiento					Ficheros Test		
Porcentaje Ficheros	Vector C	Vector G	Parámetros Optimos	Porcentaje Acierto CV	Accuracy 1	Accuracy 2	Accuracy 3
50	-5:55	-30:7	C = 4 G = -4	97,73%	6,82%	10,23%	10,80%

Tabla 5.4: Resultados de precisión. Modelo generado con el segundo subconjunto de objetos y evaluado con el primero

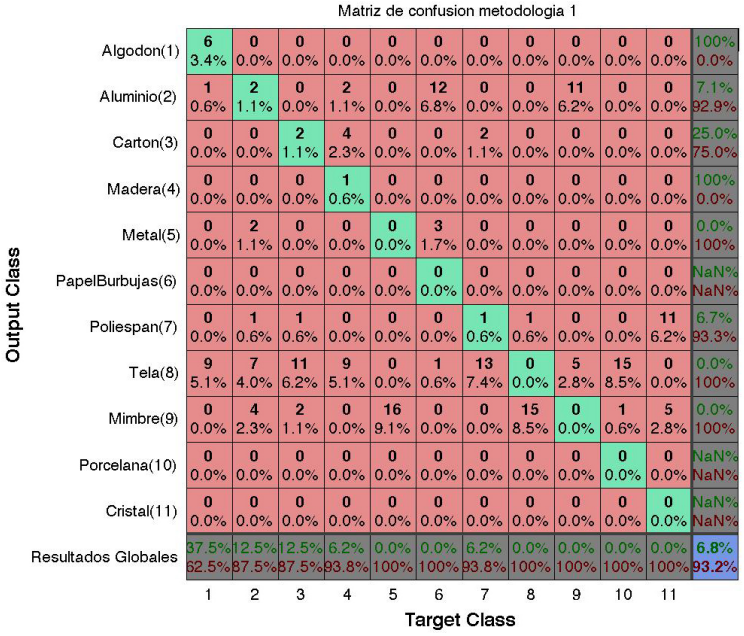


Figura 5.11: Matriz de confusión entrenando con la versión 2 (metodología 1)

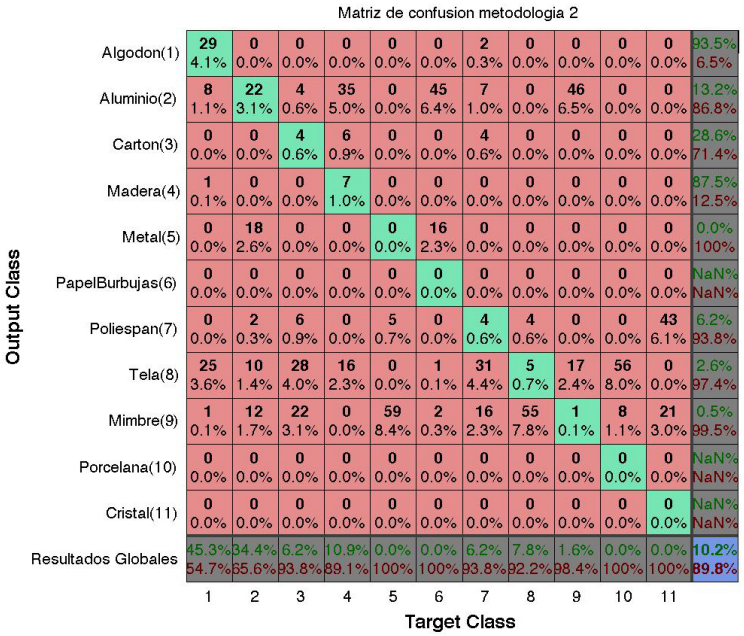


Figura 5.12: Matriz de confusión entrenando con la versión 2 (metodología 2)

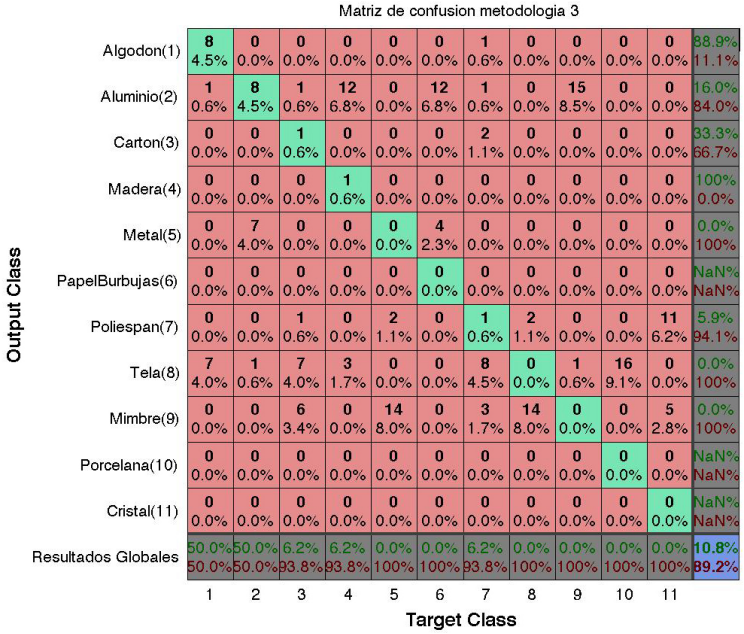


Figura 5.13: Matriz de confusión entrenando con la versión 2 (metodología 3)

5.3. Resultados cualitativos en escenas reales

En esta sección se ha analizado el uso del modelo creado en el apartado 5.2.1 para la identificación automática de materiales sobre imágenes reales complejas. Esto nos va a permitir evaluar nuestro sistema en una situación no controlada con variaciones a efectos de escala, iluminación y rotación.

Para realizar esta evaluación es necesaria la definición de tres procesos adicionales: barrido del tamaño del parche de análisis, selección del tamaño de parche que maximiza la probabilidad de cada material en cada pixel y selección de la clasificación de material para cada pixel.

5.3.1. Tamaño del parche: definición de instancia

El modelo de conocimiento generado está basado en una serie de selección de tamaños de parches definidos mediante el proceso descrito en la sección 3.4. En una imagen real con variaciones de escala (los objetos pueden estar más lejos o más cerca de la cámara) no es posible seleccionar el tamaño de parche que mejor aísle el material. Para compensar esta situación, en este trabajo se ha propuesto la selección de instancias mediante el análisis de un conjunto amplio de tamaños de parche. En este

caso se propone analizar el sistema mediante una serie de hipótesis que van desde un tamaño máximo de parche de 120x120 a un tamaño mínimo de 5x5, decrementando el tamaño en pasos de 1 pixel. De esta forma estaremos probando los sucesivos tamaños de parche y, teóricamente, tendremos más oportunidades de elegir el óptimo para cada material. Para agilizar el rendimiento del sistema, se propone barrer la imagen para cada tamaño de parche mediante un barrido no solapado .

5.3.2. Tamaño del parche: selección del tamaño óptimo

El proceso de barrido devuelve, para cada pixel de la imagen, una puntuación para cada clase y cada tamaño de parche utilizado. Primero se seleccionará para cada pixel y cada clase de material el tamaño de parche óptimo. Para ello, se seleccionará el tamaño de parche que maximiza la probabilidad devuelta por la SVM. Digamos que de esta forma tendremos un conjunto de once imágenes de probabilidades que se corresponderá cada una de ellas con las probabilidades más altas para cada pixel de pertenecer a cada una de las once clases de materiales analizadas.

5.3.3. Selección de el material en cada pixel

Finalmente, de cada una de las hipótesis de asignación de clase que representan las once imágenes mencionadas, etiquetaremos un pixel con la clase de material que maximiza la probabilidad devuelta por la SVM para cada clase. Los resultados obtenidos mediante este procedimiento se ilustran para tres casos diferentes en las figuras 5.14, 5.15 y 5.16 (etiquetado y probabilidad final de asignación) y en las figuras 5.17, 5.18 y 5.19 aislando la detección de cada material por separado.

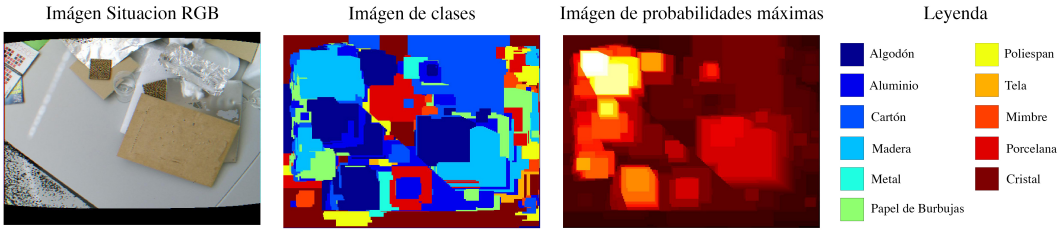


Figura 5.14: Etiquetado de clase para cada pixel y probabilidad asociada para el escenario 1

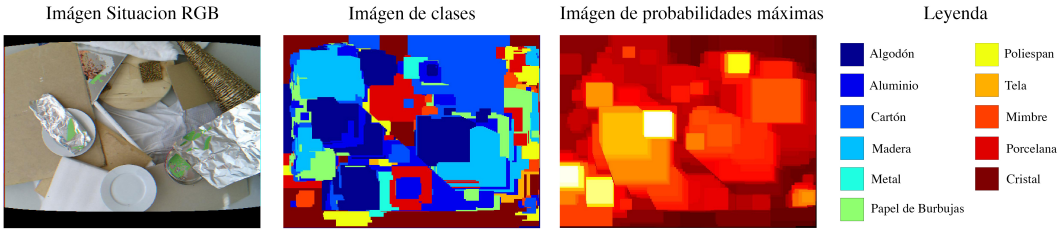


Figura 5.15: Etiquetado de clase para cada pixel y probabilidad asociada para escenario 2

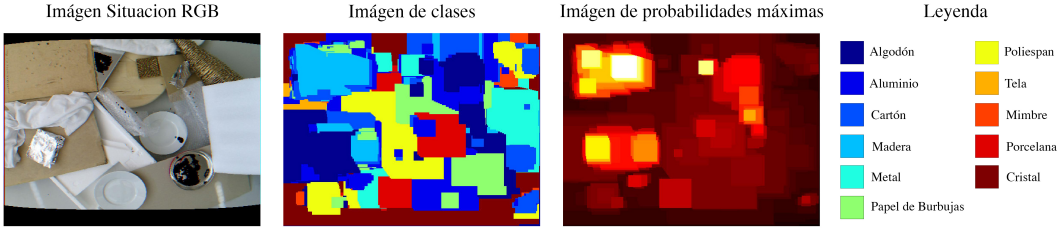


Figura 5.16: Etiquetado de clase para cada pixel y probabilidad asociada para escenario 3

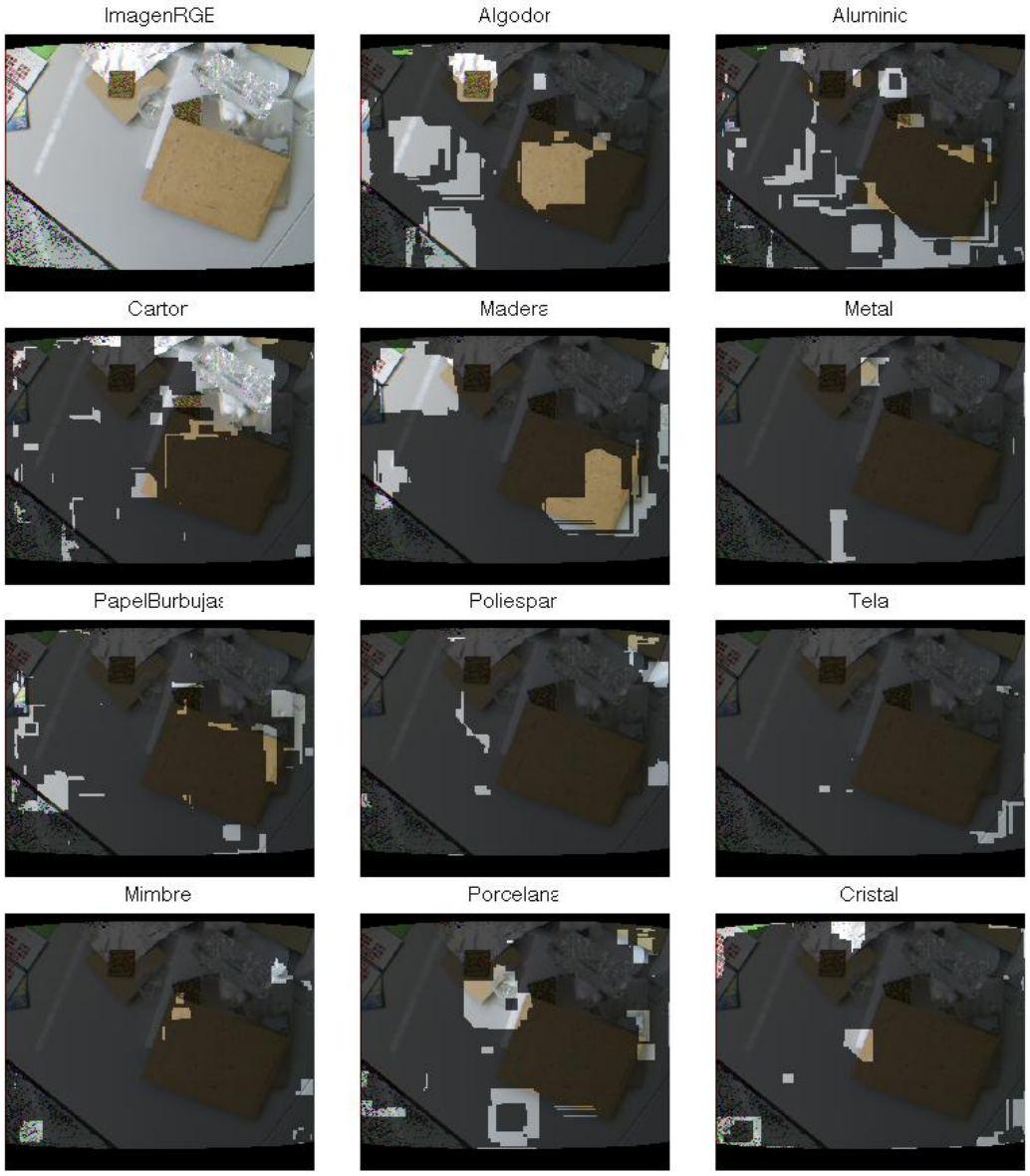


Figura 5.17: Zonas de reconocimiento de materiales para la situación 2 de la versión 1 de grabación

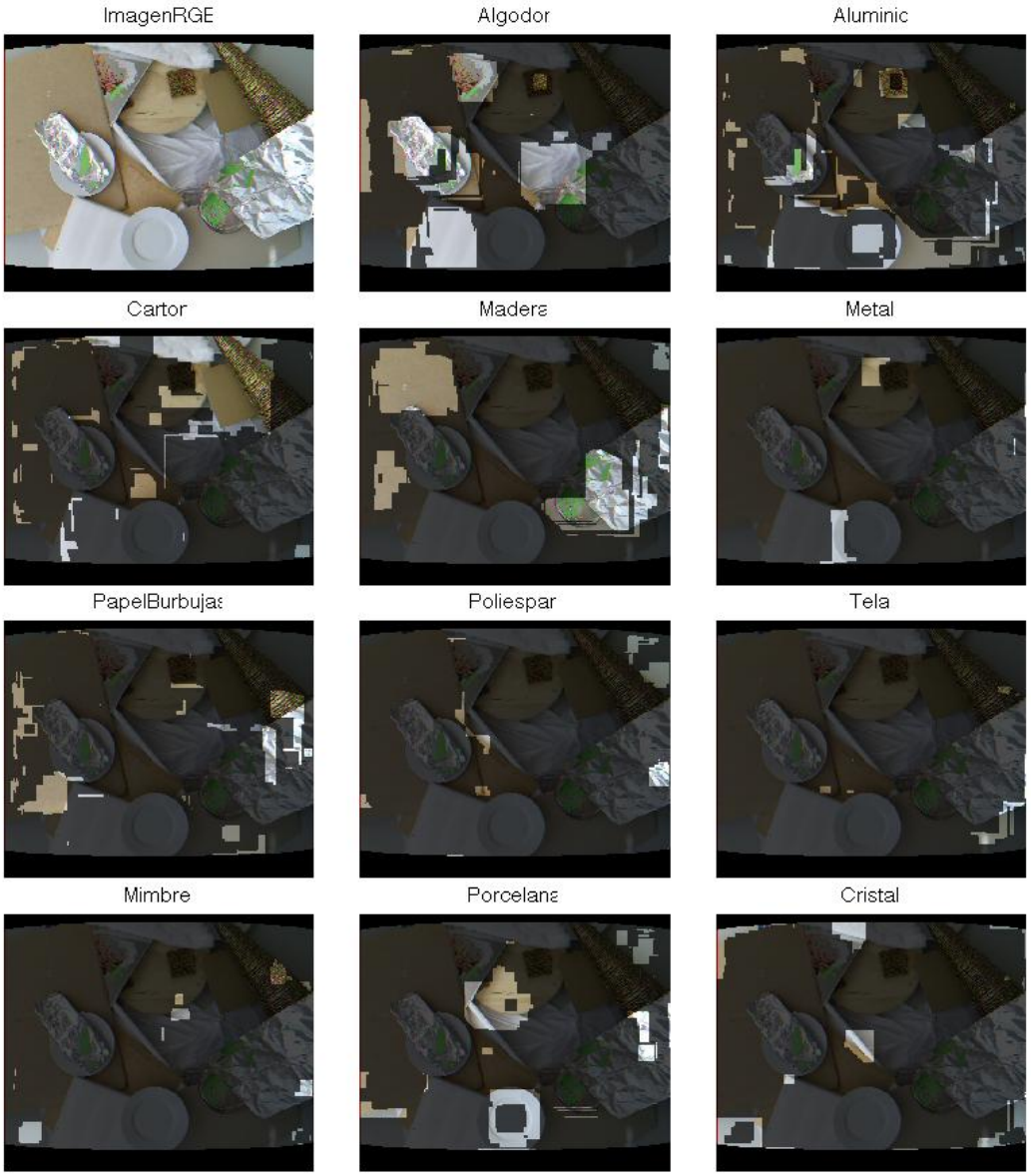


Figura 5.18: Zonas de reconocimiento de materiales para la situación 1 de la versión 2 de grabación

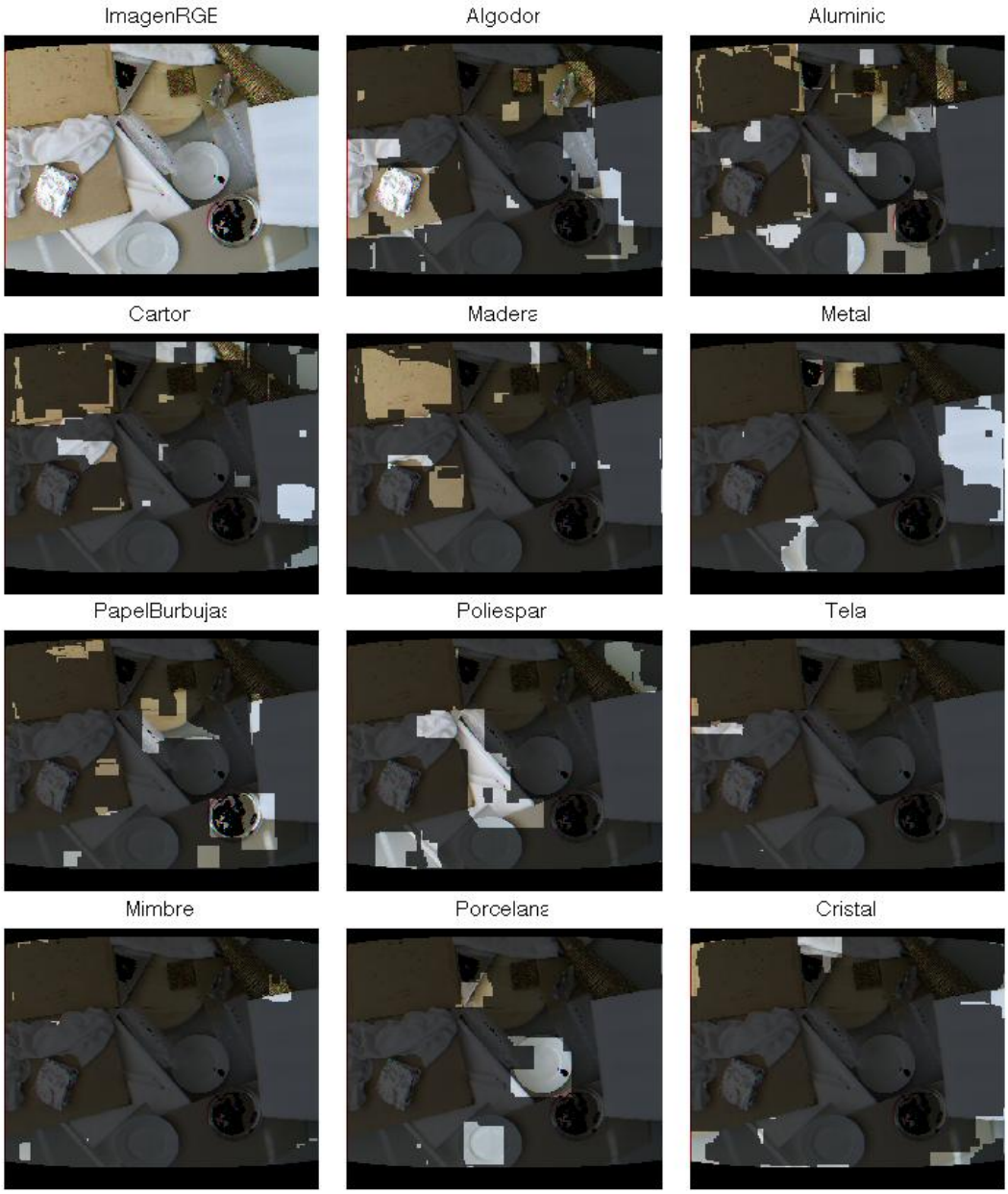


Figura 5.19: Zonas de reconocimiento de materiales para la situación 2 de la versión 2 de grabación

5.4. Discusión de resultados

El primer resultado que comentaremos será el obtenido mediante el uso completo del dataset. El proceso de entrenamiento del modelo mediante validación cruzada y el uso de dos objetos diferentes del mismo material en la grabación de la base de datos evita aparentemente el sobreentrenamiento del sistema. Esta afirmación la podemos realizar apoyándonos en los datos de precisión obtenidos mediante las tres metodologías descritas y reflejados en la figura 5.1 y en la tabla 5.1. Se evidencia que tenemos un porcentaje de precisión de validación cruzada de 85,8 % lo cual podría suponer una sobre adaptación del modelo a los datos de entrenamiento. Sin embargo, observando los valores de Accuracy 1 (85,23 %), Accuracy 2 (66,48 %) y Accuracy 3 (84,66 %) podemos comprobar que se obtienen resultados de identificación moderadamente altos y que por tanto nuestro modelo generado esta funcionando correctamente en las instancias de evaluación.

La razón de las diferencias presentes en los tres valores de precisión va a estar evidenciada en la propia definición de los mismos. Accuracy 1 va a ser el porcentaje de precisión fundamental pues va a evaluar la generación del modelo sobre los datos de entrenamiento del dataset originales, sin ningún añadido. Accuracy 2 es lógicamente el valor más bajo de los tres calculados puesto que estamos contabilizando cada fichero, incluidas las reflexiones, como una muestra independiente. Accuracy 3 va ser similar a Accuracy 1 puesto que va a evaluar teniendo en cuenta los ficheros que se usan para evaluar la metodología 1. El punto de diferencia que obtenemos entre Accuracy 1 y Accuracy 3 se debe a que hay muestras adicionales que se están clasificando como materiales erróneos con porcentajes más altos que los que obtienen los parches originales (evaluados en la metodología 1). El hecho por tanto de que las muestras originales se clasifiquen correctamente pero obteniendo probabilidades mas bajas que las muestras adicionales hace que la clase final asignada sea la correspondiente a la muestra adicional convirtiendo de esta forma la instancia original en errónea cuando en realidad esta bien clasificada (en la metodología de evaluación 3). Esto nos puede dar una idea sobre si nuestros descriptores usados son realmente invariantes a rotación puesto que de ser así no deberían de generar los errores comentados.

Las tasas moderadas de acierto globales prueban que nuestro sistema de reconocimiento esta operando bien, sin embargo, para conocer la naturaleza de los fallos debemos de conocer los porcentajes de error de cada material independientemente. Este dato es importante puesto que nos dará información que podremos utilizar para obtener mejores porcentajes de acierto globales mejorando la identificación de aquellos materiales que no estén funcionando correctamente. Las instancias que no están

siendo identificadas correctamente se reflejan en las matrices de confusión (figuras 5.2, 5.3 y 5.4). Si observamos las celdas coloreadas de rojo obtendremos el numero de ficheros correspondientes de un material que se han identificado incorrectamente y, lo que es más importante, cual ha sido el material erróneo asignado.

Valorando estas matrices comprobamos que el problema esta fundamentalmente en el cartón y la tela. Muchas muestras correspondientes al cartón se están asignando de forma errónea a la tela. Esta reflexión se ve reflejada en el numero de muestras de cartón identificadas como tela de las matrices de confusión (5 para Accuracy 1, 24 para Accuracy 2 y 6 para Accuracy 3) Esto puede estar provocado porque las características de ambos materiales sean similares en el conjunto de objetos grabado. Estos materiales pueden tener una distribución de la luz incidente similar que hace que la descripción por intensidades y orientaciones de gradientes no sea completamente descriptiva de los dos, no los hagan completamente independientes y se puedan confundir. De igual forma tenemos otros materiales como el aluminio y el mimbre que dan algún problema pero en menor medida que referentes a los materiales comentados anteriormente.

Si comentamos los resultados relacionados con el aumento del numero de ficheros de entrenamiento utilizados para evaluar el modelo se evidencia que la validación cruzada esta funcionando correctamente y que evita aparentemente el sobreentrenamiento del sistema. Los porcentajes de acierto de test aumentan a medida que estamos utilizando más ficheros de la base de datos para entrenar, situación completamente contraria a la que se produciría si tuviéramos un modelo sobreentrenado. Esta afirmación esta apoyada en los resultados obtenidos para las tres metodología de evaluación utilizadas de la tabla 5.2 representados gráficamente en la figura 5.5.

Hay que destacar que esto es en gran medida aparte del uso de la validación cruzada, gracias a la buena elección del par de parámetros (C, G). Pese a las altas tasas de reconocimiento obtenidas mediante el aumento del porcentaje de ficheros de entrenamiento en este trabajo no se considera buena opción superar un umbral entorno al 60 % de ficheros de entrenamiento puesto que las medidas de evaluación podrían no ser correctas al estar destinando una cantidad baja de muestras al test del modelo.

Si pasamos ahora a valorar los resultados obtenidos en la sección 5.2.4 podemos concluir que la decisión de grabar dos conjuntos de materiales diferentes fue acertada ya que nos va a permitir generar un modelo de conocimiento capaz de modelar información de materiales y no de objetos. Si únicamente modelamos usando una versión del dataset de objetos estaremos generando un modelo capaz de identificar esos mismos objetos, sin embargo cuando introduzcamos nuevas muestras de los mismos materiales presentes en objetos diferentes el sistema no funciona correctamente.

Esta conclusión esta respaldada por los datos de precisión obtenidos tanto para el conjunto completo del dataset, como los obtenidos en las dos pruebas realizadas generando los modelos con una sola versión de la base de datos de materiales.

Tal y como vemos en las tablas 5.3 y 5.4 en ambos casos obtenemos un alto porcentaje de reconocimiento en Validación Cruzada (93,75 % para el primer caso y 97,73 % para el segundo). Esto es debido a que estamos utilizando el mismo objeto para modelar un material en el modelo de conocimiento y estaremos creando una frontera de decisión en la SVM mucho más concreta para este material. Este hecho va a provocar que los errores se reduzcan y que aumentemos considerablemente el porcentaje de acierto en validación cruzada. Sin embargo, el hecho de tener altas tasas de acierto no asegura que el reconocimiento en el otro subconjunto de objetos vaya a ser alto. Como podemos comprobar en las diferentes tasas de Accuracy de ambas tablas en ningún caso se supera el 25 % obtenido para el primer modelo e incluso llegamos a tener tasas de acierto de 6,82 % y 10,80 % en el segundo caso. Los porcentajes de precisión por tanto son realmente bajos comparados por ejemplo con los obtenidos usando el 50 % de la base de datos completa para entrenamiento (tabla 5.1).

En este caso si nos fijamos en las matrices de confusión tanto de las figuras 5.14, 5.15 y 5.16 como de las figuras 5.11, 5.12 y 5.13 podemos visualizar de otro modo la tónica comentada anteriormente. Excepto materiales como el algodón y aluminio que obtienen unos porcentajes cercanos al 10 % y 3 % respectivamente (realmente bajos comparados con el primer resultado) el resto no supera el 1 % constatando la ineficacia total de los modelos generados.

Por último en la sección 5.3 se han evaluado tres imágenes reales obteniendo resultados prometedores. Ciertos materiales como la madera, la porcelana, el mimbre e incluso el metal se identifican de forma correcta en algunas zonas de las imágenes evaluadas lo que concluye que el sistema pese a estar alejado de la perfección da muestras de correcto funcionamiento futuro. Estas conclusiones se pueden comprobar en las imágenes de 5.3.3. Estos resultados obtenidos están motivados por los problemas que presenta el análisis de una imagen compleja. En estas imágenes tenemos una serie de factores como cambios de escala, rotaciones, oclusiones de materiales, indeterminación de tamaños de parche.. etc que no están contemplados en el entrenamiento y que por tanto hacen que el funcionamiento del sistema sobre estas imágenes no sea correcto.

Evidentemente este trabajo desarrolla un prototipo de reconocedor, pero estos resultados sientan una base para corroborar que podría ser posible la identificación de esta serie de materiales en un futuro.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Los objetivos principales del proyecto eran tanto diseñar y capturar una base de datos como el diseño de un sistema capaz de identificar materiales de forma automática. Para ello se analizo el estado del arte actual en el campo de investigación comentado. Se analizaron las diferentes versiones *hardware* de Kinect definiéndola como la mejor opción para el trabajo.

Igualmente se comprobó la escasa existencia de descriptores que permitieran realizar un análisis optimo de superficies de materiales existiendo únicamente aquellos que se definían como aproximaciones tanto bidimensionales como tridimensionales a la *Bidirectional Reflectance Distribution Function* (BRDF). Algunos de ellos como GDF-HoG y EigHess-HoG se definían en su literatura como invariantes a rotaciones, característica que como se ha podido comprobar en la implementación realizada a lo largo de este trabajo y en concreto en los resultados no es completamente cierta. De igual forma se analizo como método de clasificación las *Support Vector Machines* (SVM) siendo elegido como el método optimo para la aplicación en el sistema.

Por ultimo pero no menos importante, en el análisis del estado del arte se observó que había una carencia de bases de datos que incluyeran capturas de imágenes de profundidad por lo que se decidió trabajar sobre este aspecto a lo largo del trabajo mediante la generación de la base de datos comentada anteriormente.

Se decidió, por tanto, grabar una base de datos de una selección de materiales desde diversos ángulos de captura e iluminaciones así como variaciones en la naturaleza de esta ultima (luz artificial y luz natural). Este conjunto de imágenes se capturo mediante Kinect debido primero al análisis desarrollado en el estado del arte como sensor y segundo a su capacidad de obtención de imágenes en profundidad eliminando

así la carencia de las bases de datos disponibles actualmente.

Adicionalmente se desarrollo un procedimiento por el cual poder aislar el material grabado de su entorno para su correcta descripción. De esta forma se selecciona un área espacial determinada de la captura del material y se elimina la influencia de las descripciones que lo rodean y que no tienen ningún valor para el sistema planteado.

Se ha desarrollado un sistema que toma como entrada esta base de datos, en concreto los parches generados de los materiales, y cuya funcionalidad es describir de forma optima sus características. Para ello se ha hecho uso de GDF-HoG y EigHess-HoG sobre cuatro tipos de imágenes(color, profundidad, profundidad promediada e infrarrojos) para así caracterizar un material en un determinado angulo y con una determinada iluminación.

De la misma forma se ha incluido en el sistema las SVM que nos van a permitir partiendo de las descripciones de cada material generar un modelo de conocimiento que servirá para clasificar instancias de evaluación. Esta clasificación se ha evaluado sobre instancias de entrenamiento obtenidas de la división aleatoria de la base de datos o bien del análisis de situaciones reales de conjuntos complejos de materiales en la misma imagen.

Los resultados sobre instancias de test capturadas en las mismas condiciones que las de entrenamiento son realmente prometedores dado que se obtienen altos porcentajes de acierto utilizando la base de datos completa. Hay materiales que pese a estas tasas elevadas no se clasifican de forma correcta y que puede deberse a una mala elección de los parches en la base de datos, a la ineficacia de GDF-HoG y EigHess-HoGs para describir de forma discriminativa sus superficies o bien porque el modelo no es capaz de modelar sus características.

Sobre imágenes reales también se obtienen resultados esperanzadores aunque a otro nivel que los obtenidos sobre instancias de entrenamiento. El sistema desarrollado consigue clasificar algunas zonas de diversas situaciones de forma compleja permitiendo conocer el material que esta presente en esa zona de la imagen. El funcionamiento incorrecto del sistema sobre estas imágenes viene condicionado por los cambios de escala y rotaciones en este tipo de imágenes al igual que las oclusiones presentes en los materiales, factores todos ellos imposibles de modelar mediante nuestros datos de entrenamiento y que hacen que el rendimiento del sistema en estas situaciones decaiga.

6.2. Trabajo futuro

Si observamos tanto el estado del arte actual así como los resultados obtenidos y las conclusiones extraídas podemos sentar las lineas de trabajo futuro.

En primer lugar hemos determinado que para la generación de un modelo de conocimiento basado en materiales y no en objetos es importante contar con varias instancias del mismo material presentes en diferentes objetos de la vida real, por lo tanto se pueden incluir nuevas grabaciones en la base de datos capturada manteniendo el numero de materiales o aumentándolo pero sobre todo añadiendo grabaciones de nuevos objetos. De esta forma estaremos aumentando la funcionalidad del sistema para poder identificar mejor materiales de forma automática.

Por otro lado en el anexo 7.1 se explica un nuevo descriptor desarrollado por [15] que promete describir de forma realmente precisa una superficie de un material. Este descriptor no se ha incluido en el trabajo por no estar su literatura oficialmente aceptada sin embargo su uso en un futuro (siempre y cuando se cumplan las expectativas) puede generar un aumento considerable en la precisión del sistema por lo tanto se considera como una de las ramas de trabajo a tener en cuenta en el futuro.

Adicionalmente se propone un nuevo método de análisis para obtener nueva información sobre las imágenes reales de situaciones complejas. En el trabajo se utilizo un barrido no solapado mediante el tamaño del parche para agilizar el rendimiento del sistema. En un futuro este barrido se puede llevar a cabo solapando los parches y obteniendo así nuevas probabilidades para cada pixel que nos permitirá obtener información capítulo precisa sobre las clases asociadas a cada uno de ellos.

De igual forma se contempla la sustitución de la cámara actual utilizada por cualquier revisión del mismo modelo u otra diferente y que nos permita obtener imágenes tanto de color como de profundidad con mejor calidad ya que esto se traducirá automáticamente en un aumento del porcentaje de eficacia del sistema.

Capítulo 7

Anexos

7.1. Anexo 1. LOAD

El calculo del descriptor LOAD según se propone se divide en tres diferentes etapas que se detallan a continuación:

1. Descripción de punto

   Se parte de una serie de parches circulares obtenidos de cualquier base de datos bajo diferentes rotaciones. El objetivo como se ha comentado es diseñar un descriptor invariante a estas rotaciones. Habitualmente el proceso es estimar una orientación principal y después alinear el parche en función de esta orientación, sin embargo en este caso esta dirección principal no es necesario calcularla. En particular, partiendo de una imagen puede generarse un parche circular alrededor de un punto O. Una vez obtenido este parche y dado que es simétrico para cualquier diámetro posible que se dibuje, podemos rotarlo pudiendo obtener así infinitos parches con un ángulo de rotación arbitrario.

   Para cada punto A dentro del parche circular se puede establecer un sistema de coordenadas adaptativo (ACS) que trata de seleccionar unos puntos alrededor de Ade tal forma que A y sus vecinos tengan una relación de invariancia a la rotación. Esto se consigue debido a que la posición de los vecinos de A con respecto a A es siempre la misma independientemente de la rotación que haya sufrido el parche. Podemos ver este proceso gráficamente en la figura 7.1

   El siguiente paso es codificar el patrón obtenido para el punto A, y para ello los autores proponen la siguiente codificación:

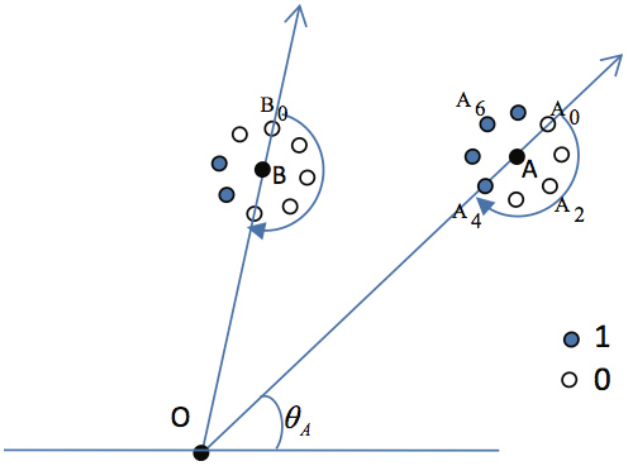


Figura 7.1: Sistema de coordenadas adaptativo (ACS) a partir del punto O y A

$$LOAD_{P,R}\left(x_A, y_A, \theta_A\right) = \sum_{p=0}^{P-1} sign(V(A_p) - V\left(A\right))2^p \tag{7.1}$$

$$\begin{cases} x_{A_p} & = x_A + Rcos\left(2\pi/P - \theta_A\right) \\ y_{A_p} & = y_A - Rsin\left(2\pi/P - \theta_A\right) \end{cases}$$

donde P es el numero de vecinos seleccionados alrededor de A (en este caso), R es el radio entre el punto $A = (x_A, y_A)$ y su vecino p-esimo en el parche, dado el sistema de coordenadas ACS: $A_P = (x_{A_P}, y_{A_P})$. $V(A)$ y $V(A_P)$ son los valores de los puntos A y A_P respectivamente, es decir, el valor del pixel A y del vecino p-esimo, y por ultimo $\theta_A = arctan\left(\frac{y_a - y_o}{x_a - x_o}\right)$, utiliza tanto la posición del punto A como del punto O inicial para calcular el ángulo que forma la recta que une ambos puntos. Este proceso devuelve una secuencia de unos y ceros que permite codificar un vecindario de pixeles. En el caso de ejemplo para A en la figura 7.1 la secuencia es "00001111".

De la misma forma en ACS y dados los mismos puntos se puede calcular la magnitud adaptativa del gradiente de la siguiente forma:

$$M(A) = \sqrt[2]{(V(A_4) - V(A_0))^2 + (V(A_6) - V(A_2))^2} \tag{7.2}$$

con la condición de que $M(A)$ solo se va a calcular cuando el radio sea $R = 1$.

Esta forma de codificar un vecindario de puntos según la ecuación 7.1 presenta varias ventajas. La primera es que va a ser invariante a rotaciones puesto que

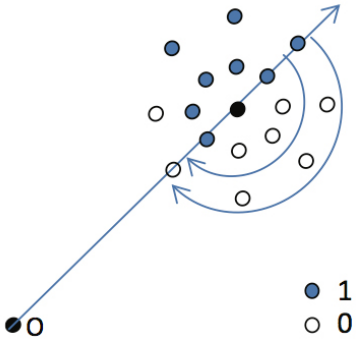


Figura 7.2: Calculo de LOAD en diferentes escalas

la relación entre un punto y su vecindario va a quedar fijada, como podemos ver en la figura 7.1 por lo que el punto A_0 siempre va a tener la misma posición relativa con respecto al punto de origen A independientemente de la rotación del parche, puesto que A_0 se selecciona sobre la recta que une O con A en el punto más alejado de ésta en el que la recta corte a la circunferencia con centro en A y radio $R = 1$. Además el descriptor es invariante a cambios de iluminación globales del parche. p

2. Descripción multi-escala

   El análisis multi-escala es una buena aproximación para representar información de textura en diferentes escala. Estructuras que a lo mejor están correctamente definidas en una escala mayor pueden no estarlo en una menor y viceversa por lo tanto se propone como ampliación añadir al diseño del descriptor LOAD la variable de la escala. Para ello se propone la reformulación de la ecuación 7.1 de la siguiente forma:

$$LOAD_{P,R}\left(x_A, y_A, \theta_A, s\right) = \sum_{p=0}^{P-1} sign(V(A_p) - V\left(A\right))2^p \tag{7.3}$$

$$\begin{cases} x_{A_p} & = x_A + s \times Rcos\left(2\pi/P - \theta_A\right) \\ y_{A_p} & = y_A - s \times Rsin\left(2\pi/P - \theta_A\right) \end{cases}$$

   donde s es el factor de escala analizado. De esta forma pueden obtenerse patrones de LOAD en diferentes escalas de la imagen. Esto se representa gráficamente en la figura 7.2

   Como podemos observar tenemos dos conjuntos de vecinos a dos escalas diferentes. Usando la ecuación 7.3 podemos calcular que ambos patrones están

codificados con los códigos “10001111” y “10000011”. Como ambos patrones van a describir un vecindario de pixeles podemos llegar a la conclusión de que la similitud entre ambos códigos (en este caso tenemos errores en dos dígitos únicamente) indicara que las estructuras alrededor del punto que genera el patrón son consistentes a lo largo de las escalas en las que se ha realizado el análisis.

3. Construcción de histograma y normalización

Si tenemos por tanto un parche circular con un punto O definido como su centro podemos suponer que el parche tiene K puntos. Hacemos la suposición de que trabajaremos con S escalas y ademas, conociendo de antemano que la dimensión del descriptor LOAD de cada escala S es de 59 muestras, podemos obtener que la dimensión final del descriptor será $59 \times S$.

Para comenzar el algoritmo inicializamos un histograma H que constituirá el descriptor completo y que se rellenará de la siguiente forma para cada punto O_i, $i \, \epsilon \, [1, K]$, es decir para cada punto del parche:

$$H(U_s(O_i), s) = H(U_s(O_i), s) + M(O_i) \tag{7.4}$$

donde $s \, \epsilon \, [1, S]$ es el numero de escala, $M(O_i)$ es la magnitud del gradiente del punto O_i en ACS calculada como se indica en la ecuación 7.2 y $U_s(O_i)$ es el patrón uniforme de LOAD del punto O_i en la escala s calculado según 7.3.

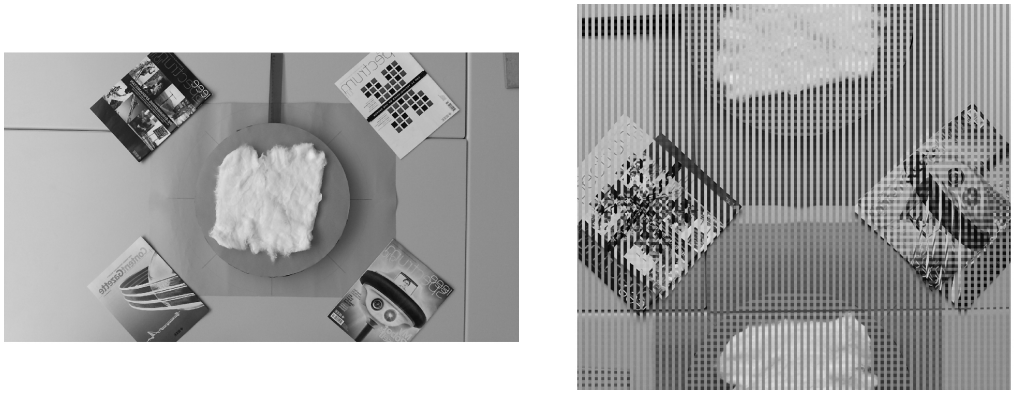
Después de tener los K puntos del parche circular acumulados en el histograma reconvertimos este en un vector de una sola dimensión. Después se propone usar una normalización de características, en este caso se utiliza la operación raíz cuadrada.

7.2. Anexo 1. Obtención de la imagen en color 1080p

1. Luminancia: Obtenida mediante el formato definido en el SDK como "Yuy2" de tamaño 1080x1920p.

2. RGB: Obtenida mediante el formato "rgba" del SDK de tamaño 2160x1920p.

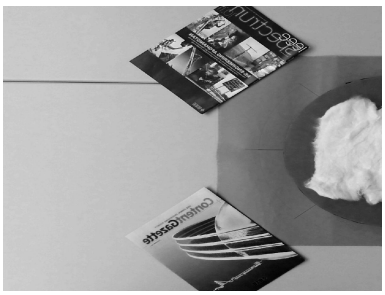
Las imágenes que obtenemos de ambos formatos son las representadas a continuación:



(a) Imagen de luminancia (b) Imagen RGB

Figura 7.3: Imágenes de luminancia y RGB

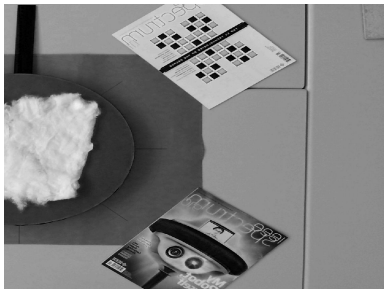
Como podemos observar la imagen RGB viene claramente mezclada y no se corresponde con ninguna imagen en color ordenada a primera vista. Sin embargo haciendo pruebas de submuestreo se pueden obtener 4 imágenes diferentes mediante el uso de Matlab.



(a) Submuestreo 1 columna de cada 2 empezando por la primera y una fila de cada 2 empezando por la primera

(b) Submuestreo 1 columna de cada 2 empezando por la segunda y una fila de cada 2 empezando por la segunda

(c) Submuestreo 1 columna de cada 2 empezando por la primera y una fila de cada 2 empezando por la segunda



(d) Submuestreo 1 columna de cada 2 empezando por la segunda y una fila de cada 2 empezando por la primera

Figura 7.4: Imágenes obtenidas tras submuestrear la imagen RGB

Por lo que si concatenamos las imágenes de la figura 7.4 de la siguiente forma:

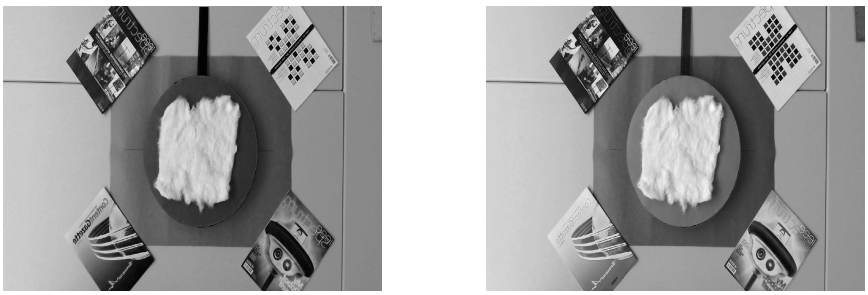
$$Azul = [Aux4', Aux2'];$$

$$Rojo = [Aux1', Aux3'];$$

Siendo

- Aux1 la imagen de la figura 7.4a
- Aux2 la imagen de la figura 7.4b
- Aux3 la imagen de la figura 7.4d
- Aux4 la imagen de la figura 7.4c

obtenemos las siguientes imágenes:



(a) Rojo (b) Azul

Figura 7.5: Imágenes Rojo y Azul

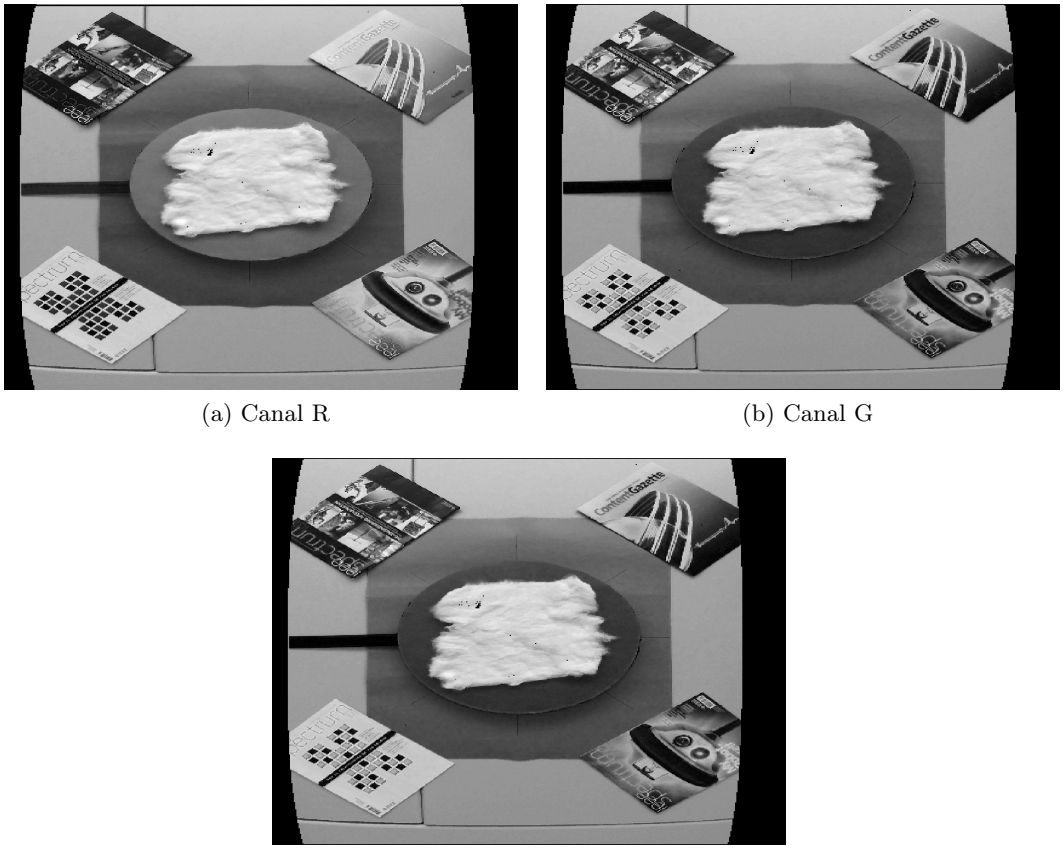
Como podemos observar estas imágenes ya tienen forma definida y por lo tanto podrían representar algún canal del formato RGB. Finalmente si juntamos estas dos imágenes con la luminancia obtenemos la figura 7.6

Figura 7.6: Imagen en color final

De esta forma conseguimos tener la imagen en color real con una resolución Full HD de 1080x1920p. En este punto nuestro programa actualizado Kinect ML junto con un procesado de Matlab será capaz de capturar de forma automática y procesar las siguientes tres imágenes.

7.3. Anexo 2. Generación de la imagen RGB alineada

Ese bloque se puede diseñar para que utilice el método de calibración de la cámara mediante los parámetros intrínsecos de la misma o bien haciendo el procedimiento básico de captura de un tablero de ajedrez para después utilizar las correspondencias para ajustar la imagen de color a la de profundidad. Sin embargo, el propio SDK utilizado para la captura de las imágenes, permite realizar este proceso mediante la función descrita como *Coordinatemapper.MapDepthFrameToColorSpace()* que va a mapear una imagen en profundidad al espacio de color.



(a) Canal R (b) Canal G

(c) Canal B

Figura 7.7: Canales RGB mapeados

Esta función va a tomar una imagen en profundidad y va a rellenar los puntos donde esta imagen tenga un valor optimo con valores correspondientes a un frame de color. De esta forma vamos a tener los valores de una imagen de color en las posiciones físicas de un frame de profundidad todo ello divido ademas en tres canales RGB. Estos canales RGB ante la imposibilidad de ser guardados en un fichero .mat

Figura 7.8: Frame color RGB mapeado

son guardados en crudo en ficheros binarios hexadecimales que posteriormente serán leídos con Matlab. Estos canales están representados en la figura 7.7.

Por tanto partiendo de los tres canales se puede generar una imagen correcta en RGB que podemos visualizar frente a la profundidad e infrarrojos en la figura 7.8.

7.4. Anexo 3. Librería LibSVM

En este anexo se comentan varias configuraciones de diferentes aspectos que podemos realizar mediante la librería.

1. Tipo de SVM:

   La librería nos permite variar el tipo de maquina que queremos utilizar dependiendo de la situación que mas se adapte. Se definen hasta cinco tipos diferentes de maquinas:

   a) C-SVC: Este es el tipo de SVM que viene definido por defecto y que va a basar su uso en la clasificación multiclase. C será el coste de la función y será un parámetro configurable.

   b) nu-SVC: Del mismo modo que C-SVC vamos a tener este modo para generar un modelo de clasificación con diferentes tipos de clases. Ambas realizan básicamente la misma función pero con diferentes parámetros. En este caso el parámetro configurable será nu que tendrá un rango entre [0,1] mientras que C será desde 0 hasta infinito.

   c) one-class SVM: Como su propio nombre indica servirá para el modelo de clasificación de una sola clase, es decir, un modelo de clasificación binario de si una muestra corresponde a la clase que hemos entrenado o no.

   d) epsilon-SVR: Este modelo al contrario de los descritos anteriormente no tiene como objetivo la clasificación de una muestra si no generar una solución de un problema de regresión, es decir predecir el valor de la siguiente muestra en función de un modelo dado.

   e) nu-SVR: De nuevo se trata de un modelo de regresión que va a funcionar de manera similar a epsilon-SVR pero variando los parámetros que necesita.

2. Tipo de kernel y parámetros

   Como se vio en 2.4.2 las SVM dependen de funciones kernel que nos van a proyectar nuestros datos de entrenamiento en un espacio de una dimensión mayor. Por esta razón la selección de la función de kernel va a ser un paso importante y la librería nos da varias opciones:

   a) Lineal

   b) Polinomial

   c) Función de base radial (Radial Basis Function)

   d) Sigmoid

e) Kernel precomputado

El método de entrenamiento y test seleccionado para el sistema son las maquinas de vector soporte. Para la implementación de las SVM se ha utilizado la librería LIBSVM para Matlab disponible gratuitamente en Internet [27]. Esta librería contiene un paquete de varias funciones que nos van a permitir realizar tanto el modelo mediante unos datos de entrada como predecir una nueva muestra.

Las funciones disponibles son:

$$Model = svmtrain(TrainingLabelVector,\ TrainingInstanceMatrix,\ 'Opciones'); \tag{7.5}$$

$$[PredictLabel,\ Accuracy,\ ProEstimates] = \tag{7.6}$$

$$svmpredict(TestLabelVector,\ TestInstanceMatrix,\ Model,\ 'Opciones');$$

Esto nos indica que la base de datos con la que trabajemos tendrá que estar previamente dividida en un conjunto de entrenamiento y un conjunto de test. Una vez tenemos realizada la división lo único que habrá que calcular serán los vectores $TrainingLabelVector, TrainingInstanceMatrix, TestLabelVector$ y $TestInstanceMatrix$ y podremos usar las funciones sin demasiada dificultad, sin embargo tendremos en cada una de ellas un ultimo parámetro de opciones que nos permitirá modificar cada función y que describiremos mas adelante en detalle.

Los vectores van a tener el mismo formato independientemente de si estamos trabajando con los subconjuntos de Test o Train, por lo tanto se podrá realizar una misma función que dados unos ficheros, parches en este caso de la base de datos, de entrada genere automáticamente tanto el vector de etiquetas como la matriz de vectores de características.

El formato del $LabelVector$ va a ser un vector de $(NumeroFicheros, 1)$ en el cual se etiquetará cada fichero con un numero correspondiente a su clasificación, es decir, tendremos un vector en cual cada fila corresponde a un archivo y mapearemos ese fichero a un numero según corresponda a un material u otro de la base de datos. La asignación para realizar el mapeo es la siguiente:

1. Algodón
2. Aluminio
3. Carton

4. Madera
5. Metal
6. Papel de burbujas
7. Poliespan
8. Tela
9. Mimbre
10. Porcelana
11. Cristal

Este vector estará presente tanto en la función de generación del modelo como la de predicción de una muestra, sin embargo en esta ultima podremos introducir un vector de números aleatorios dado que lo normal es que cuando tratemos con ficheros de testeo de un modelo no tengamos la información de a que categoría corresponde a cada uno, puesto que ese es precisamente el trabajo del clasificador y lo que estamos buscando.

En cuanto a $InstanceMatrix$ se tratara de una matriz de dimensión *NumeroFicheros, NumerodeMuestras* en la cual introduciremos en cada fila el vector de características correspondiente a ese fichero. Evidentemente el mapeo de ficheros tanto en $LabelVector$ como en $InstanceMatrix$ tiene que ser el mismo, es decir el fichero de la fila número 1 debe de ser el mismo tanto en el vector como en la matriz.

Una vez tenemos estos parámetros correctamente calculados pasamos tanto a la generación del modelo como a la prueba de rendimiento del mismo.

Esa será la etapa siguiente y que dependerá 100 % de nuestro sistema de entrenamiento. En esta parte simplemente introduciremos unos datos de testeo a partir de $TestLabelVector$ y $TestInstanceMatrix$ (como se ha comentado en la sección **??** en este caso el vector de etiquetas se puede crear mediante números aleatorios puesto que en un sistema real no vamos a saber a que etiqueta corresponde cada archivo) y utilizando $[PredictLabel, Accuracy, ProEstimates] = svmpredict()$ podremos obtener los tres parámetros devueltos.

1. $PredictLabel$

   Será un vector con las etiquetas que el modelo a determinado para cada archivo de test que le hemos introducido. Es realmente la decisión que ha tomado el modelo acerca de si una entrada es de una clase u otra.

2. *Accuracy*

   Este valor solo se devolverá correctamente si estamos en un entorno controlado y sabemos a que clase corresponde realmente cada muestra de test y hemos rellenado previa y correctamente el $TestLabelVector$. Internamente la función comparara $TestLabelVector$ y $PredictLabel$ y comprobara cuales de esas etiquetas son iguales y por tanto cual es el porcentaje de acierto.

3. *ProEstimates*

   Si configuramos para que tanto el modelo como la predicción se ejecuten en modo probabilidades nos devolverá una matriz con el numero de filas correspondiente al numero de ficheros de test y con tantas columnas como materiales tengamos en nuestro modelo. Cada valor de la columna será la probabilidad determinada por el modelo de que ese archivo sea ese material.

Como podemos observar los módulos tanto de extracción de características como de generación y testeo de modelo con todo lo que ello conlleva se han explicado tanto en la sección 4.1.1 como en la **??**. Sin embargo estos módulos no parten de la base de datos completa si no que necesitan una base de datos divida en dos subconjuntos de Train y Test. Para la realización de este proceso hay dos bloques principales:

1. Generación de una carpeta temporal

   Este es el modulo que parte de la base de datos grabada al completo. Va a modificar la estructura de la base de datos de forma que todos los archivos relacionados con un material van a estar en una carpeta con el nombre del mismo. Ya no vamos a tener separación por tipos de luces ni por ángulos si no que todos los archivos de algodón por ejemplo estarán en la misma ruta "DataSet-Temporal/Algodón". Sin embargo la información de a que luz corresponde cada archivo y a que ángulo no la vamos a perder puesto que cambiamos el nombre de cada fichero con la siguiente estructura:

   $$TipoParche\_Material\_TipoLuz\_Angulo\_versionBD.mat$$

   Este renombramiento del fichero nos va a permitir indexar los archivos por el nombre de forma cómoda sin perder ningún tipo de información al respecto de un archivo.

2. Generación de las carpetas Test y Train

   Este modulo va a tener como entrada la carpeta temporal generada anteriormente y un único parámetro, el porcentaje de ficheros de la base de datos que

definiremos como subconjunto de Train. De esta forma podremos configurar automáticamente el numero de ficheros que tendremos en este subconjunto y por lo tanto que los restantes se configuren como test.

La estructura de esta carpeta será idéntica a la de la carpeta temporal es decir:

- DataSetTT
  - ✧ Train
    - ❏ Algodón
      - ▷ *TipoParche_ Material_ TipoLuz_ Angulo_ versionBD.mat*
      - ▷ ...
      - ▷ *TipoParche_ Material_ TipoLuz_ Angulo_ versionBD.mat*
    - ❏ Aluminio
    - ❏ ...
    - ❏ Tela
  - ✧ Test

Para la selección de que ficheros van a ser asignados a Train y Test se realiza un proceso aleatorio. Sabemos por la extracción de características que tendremos un descriptor por cada angulo por lo tanto generamos aleatoriamente una selección de tantos ángulos como ficheros de Train hayamos configurado en el parámetro de porcentaje y obtendremos un numero x de ficheros cada uno con sus cuatro diferentes parches. Estos serán los que se utilizaran para el proceso de entrenamiento y los restantes para el testeo del mismo.

Bibliografía

[1] M. K. v1 For Windows, "http://www.xbox.com/es-es/xbox-360/accessories/more'#'retailerblade,"

[2] K.-T. datasets and M.-V. datasets, "https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/object-recognition-and-scene-understanding/recognizing-materials-from-virtual-examples/,"

[3] K. Hanbay, N. Alpaslan, M. F. Talu, D. Hanbay, A. Karci, and A. F. Kocamaz, "Continuous rotation invariant features for gradient-based texture classification," *Computer Vision and Image Understanding*, vol. 132, pp. 87–101, 2015.

[4] "Microsoft kinect v1 (xbox 360),"

[5] M. K. SDK, "https://www.microsoft.com/en-us/download/details.aspx?id=44561,"

[6] D. Ravi, "Kinect: the next generation of motion control," *Universita degli Studi di Catania Dipartimento di Matematica e Informatica*, 2013.

[7] K. for Windows Team, "Kinect near mode, http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx," *Microsoft*, 2012.

[8] C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz, "Exploring features in a bayesian framework for material recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 239–246, IEEE, 2010.

[9] J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system," *Biological cybernetics*, vol. 55, no. 6, pp. 367–375, 1987.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[11] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 637–645, ACM, 2006.

[12] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM transactions on graphics (TOG)*, vol. 21, no. 3, pp. 257–266, 2002.

[13] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[15] X. Qi, G. Zhao, L. Shen, Q. Li, and M. Pietikainen, "Load: Local orientation adaptive descriptor for texture and material classification," *arXiv preprint arXiv:1504.05809*, 2015.

[16] A. Albiol, D. Monzo, A. Martin, J. Sastre, and A. Albiol, "Face recognition using hog–ebgm," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1537–1543, 2008.

[17] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient hog human detection," *Signal Processing*, vol. 91, no. 4, pp. 773–781, 2011.

[18] F. E. Nicodemus, "Directional reflectance and emissivity of an opaque surface," *Applied optics*, vol. 4, no. 7, pp. 767–775, 1965. BDRF.

[19] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 9, pp. 891–906, 1991.

[20] M. P. Do Carmo and M. P. Do Carmo, *Differential geometry of curves and surfaces*, vol. 2. Prentice-hall Englewood Cliffs, 1976.

[21] J. Zhang, H. Zhao, and J. Liang, "Continuous rotation invariant local descriptors for texton dictionary-based texture classification," *Computer Vision and Image Understanding*, vol. 117, no. 1, pp. 56–75, 2013.

[22] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision–ECCV 2010*, pp. 356–369, Springer, 2010.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[24] L. Sharan, R. Rosenholtz, and E. Adelson, "Material perception: What can you see in a brief glance?," *Journal of Vision*, vol. 9, no. 8, pp. 784–784, 2009.

[25] K. V. . D. F. to .mat File Exporter Tool (Kinect ML), "http://www.codeproject.com/tips/819613/kinect-version-depth-frame-to-mat-file-exporter,"

[26] T. C. P. O. License, "Kinect v2 point cloud using vtk for visualisation," *CodeProject*, 27 Dec 2014.

[27] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[28] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.